On Probabilistic Strategies for Robot Tasks

by
Michael A. Erdmann

B.S., University of Washington
(1982)

S.M., M.I.T.
(1984)

Revised version of a thesis submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
at the
Massachusetts Institute of Technology

August, 1989

4 82

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER AI-TR 1155	2 GOVT ACCESSION NO	3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle) On Probabilistic Strategies for Robot Tasks		5 TYPE OF REPORT & PERIOD COVERED technical report 6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s) Michael Andreas Erdmann		8 CONTRACT OR GRANT NUMBER(s) N00014-86-K-0685 N00014-85-K-0124
9 PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12 REPORT DATE March 1990 13 NUMBER OF PAGES 287
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15 SECURITY CLASS. (of this report) UNCLASSIFIED 15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

None

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

Motion planning	Robotics	Automatic programming
Uncertainty	Randomization	Fine motion
Assembly	Manipulation	Probabilistic Algorithms

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

Robots must act purposefully and successfully in an uncertain world. Sensory information is inaccurate or noisy, actions may have a range of effects, and the robot's environment is only partially and imprecisely modelled. This thesis introduces active randomization by a robot, both in selecting actions to execute and in focusing on sensory information to interpret, as a basic tool for overcoming uncertainty.

(continued on back)

Block 20 continued:

An example of randomization is given by the strategy of shaking a bin containing a part in order to orient the part in a desired stable state with some high probability. Another example consists of first using reliable sensory information to bring two parts close together, then relying on short random motions to actually mate the two parts, once the part motions lie below the available sensing resolution. Further examples include tapping parts that are tightly wedged, twirling gears before trying to mesh them, and vibrating parts to facilitate a mating operation.

Randomization is seen as a primitive strategy that arises naturally in the solution of manipulation tasks. Randomization is as essential to the solution of tasks as are sensing and mechanics. An understanding of the way that randomization can facilitate task solutions is integral to the development of a theory of manipulation. Such a theory should try to explain the relationship between solvable tasks and repertoires of actions, with the aim of creating autonomous systems capable of existing in an uncertain world.

The thesis expands the existing framework for generating guaranteed strategies to include randomization as an additional operator. A special class of randomized strategies is considered in detail, namely the class of simple feedback loops. A simple feedback loop repeatedly considers only current sensed values in deciding on actions to execute in order to make progress towards task completion. When progress is not possible the feedback loop executes a randomizing motion. The thesis shows that if the average velocity of the system points towards the goal, then the system converges to the goal rapidly.

A simple feedback loop was implemented on a robot. The task consisted of inserting a peg into a hole using only position sensing and randomization. The implementation demonstrated the usefulness of randomization in solving a task for which sensory information was poor.

This report is a revised version of a thesis submitted on August 21, 1989 to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by the Office of Naval Research under the University Research Initiative Program through contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract N00014-85-K-0124. This research was further supported in part by an NSF Presidential Young Investigator Award to Tomás Lozano-Pérez, in part by a fellowship from General Motors Research Laboratories, and in part by a fellowship from NASA's Jet Propulsion Laboratory.

On Probabilistic Strategies for Robot Tasks

by
Michael A. Erdmann

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Robots must act purposefully and successfully in an uncertain world. Sensory information is inaccurate or noisy, actions may have a range of effects, and the robot's environment is only partially and imprecisely modelled. This thesis introduces active randomization by a robot, both in selecting actions to execute and in focusing on sensory information to interpret, as a basic tool for overcoming uncertainty.

An example of randomization is given by the strategy of shaking a bin containing a part in order to orient the part in a desired stable state with some high probability. Another example consists of first using reliable sensory information to bring two parts close together, then relying on short random motions to actually mate the two parts, once the part motions lie below the available sensing resolution. Further examples include tapping parts that are tightly wedged, twirling gears before trying to mesh them, and vibrating parts to facilitate a mating operation.

Randomization is seen as a primitive strategy that arises naturally in the solution of manipulation tasks. Randomization is as essential to the solution of tasks as are sensing and mechanics. An understanding of the way that randomization can facilitate task solutions is integral to the development of a theory of manipulation. Such a theory should try to explain the relationship between solvable tasks and repertoires of actions, with the aim of creating autonomous systems capable of existing in an uncertain world.

The thesis expands the existing framework for generating guaranteed strategies to include randomization as an additional operator. A special class of randomized strategies is considered in detail, namely the class of simple feedback loops. A simple feedback loop repeatedly considers only current sensed values in deciding on actions to execute in order to make progress towards task completion. When progress is not possible the feedback loop executes a randomizing motion. The thesis shows that if the average velocity of the system points towards the goal, then the system converges to the goal rapidly.

A simple feedback loop was implemented on a robot. The task consisted of inserting a peg into a hole using only position sensing and randomization. The implementation demonstrated the usefulness of randomization in solving a task for which sensory information was poor.

Thesis Supervisor: Tomás Lozano-Pérez
Title: Associate Professor of
Electrical Engineering and Computer Science

Acknowledgments

My sincere thanks and gratitude go to my advisor Tomás Lozano-Pérez for his enormous support, encouragement, and, above all, intellectual stimulation, over the past seven years. Tomás was always there when I needed to talk to him. Many hours were spent exchanging ideas in Tomás' office. That was fun and rewarding, and I will miss the interaction. Many ideas in this thesis were born during those discussions, while others were gracefully retired. Thank you, Tomás.

Many thanks as well to the other members of my committee, Rod Brooks, Eric Grimson, and Matt Mason, for their suggestions and advice, and for detailed comments on the thesis. I very much appreciate all the reading. Thanks also to Eric for much useful advice on various occasions in the past few years. Thanks to Rod for chatting about robotics and life, and for running. Thanks to Matt for his support and for all the discussions on friction, planning, and a trillion other things. Thank you all for all you have given me.

I have very much enjoyed this process, and I thank all members of the AI Lab community for their friendliness and camaraderie. The combination of intellectual curiosity and personal warmth present in this lab is something to be treasured.

Many thanks to Bruce Donald, friend and scholar. You too were always there for me when I needed you. Thanks for all the discussions on robotics and life. Thanks for the oral exam preparations, for practice talks, for champagne. Many of the ideas in this and the preceding thesis benefited greatly from discussions with Bruce. Thanks.

Thanks to my officemates: Thanks first to John Canny and Bruce Donald, my officemates for five years. It was fun going through this process together. Thanks for all the discussions and debates on the whiteboards. I learned a lot from both of you. More recently, thanks to Michael Brent and Maja Mataric. It was fun, and I'll miss you. Thanks in particular to Maja for long hours of talking, walking, and some running. I enjoyed our talks on mobile robots and the debates on theory versus practice. Thanks for all your help.

Thanks to other members of the lab: Thanks to Mike Caine for discussions on robotics, and for the use of all his peg-in-hole assemblies. Various experiments, including those in this thesis, were conducted using those assemblies. Thanks to Mike Kashket, Jeff Koechling, and David Siegel for checking up on me, and keeping the proper perspective. Thanks to Karen Sarachik for talks and espresso. Thanks to Sundar Narasimhan for much advice on robotics and life. I couldn't have done it without you.

Thanks to Randy Brost, Ken Goldberg, Yu Wang, and Brad White at CMU for discussions and support. You made my visits to Pittsburgh particularly exciting and fun.

Thanks to the System Development Foundation, the National Science Foundation, General Motors Research Laboratories, and NASA-JPL for financial support.

Many thanks to Laura Radin for her friendship and support over the past several years.

Thanks to Andrew Ajemian, Leola Alfonso, David Clark, Bruce McDermott, and Laura Nugent for support at various times. Your friendships have been wonderful. Thanks in particular to Bruce for constant moral support and understanding advice over the last two decades. Thanks to Laura for those long runs, and selfless insights. And thanks to Andrew and Laura for Gloucester, hibachis, and the "K" episode.

Finally, thanks to my family for their never ending support. Thanks to my parents, Joachim and Ursula Erdmann, for their strong belief in education, and their advice on all matters. Thanks to my siblings, Thomas and Mariele, for being there. This thesis is dedicated to the memory of my grandmother, Martha Wedemeyer, for the example she set.

Detailed Abstract

Robots must act purposefully and successfully in an uncertain world. Sensory information is inaccurate or noisy, actions may have a range of effects, and the robot's environment is only partially and imprecisely modelled. This thesis introduces active randomization by a robot, both in selecting actions to execute and in focusing on sensory information to interpret, as a basic tool for overcoming uncertainty.

An example of randomization is given by the strategy of shaking a bin containing a part in order to orient the part in a desired stable state with some high probability. Another example consists of first using reliable sensory information to bring two parts close together, then relying on short random motions to actually mate the two parts, once the part motions lie below the available sensing resolution. Further examples include tapping parts that are tightly wedged, twirling gears before trying to mesh them, and vibrating parts to facilitate a mating operation. Randomization is also useful for mobile robot navigation and as a means of guiding the design process.

Over the past several years a planning methodology [LMT] has evolved for synthesizing strategies that are guaranteed to solve robot tasks in the presence of uncertainty. Traditionally such strategies make judicious use of sensing and task mechanics, in conjunct with the maintenance of past sensory information and the prediction of future behavior, in order to overcome uncertainty. There are two restrictions on the generality of this approach. First, not all tasks admit to guaranteed solutions. Uncertainty simply may be too great to guarantee task success in a specific number of steps. Second, a strategy is only as good as is the validity of its assumptions. In an uncertain world all assumptions are subject to uncertainty. For instance, there may be unmodelled parameters that govern the behavior of a system. This fundamental uncertainty limits the guarantees that one can expect from any strategy.

The randomization approach proposed in this thesis attempts to bridge these difficulties. First, the underlying philosophy of a randomized strategy assumes that several attempts may need to be made at solving a task. A task is only assumed to be solvable with some probability on any given attempt. This view of a solution to a task broadens the class of solvable tasks. Second, by actively randomizing its actions a system can blur the significance of unmodelled or uncertain parameters. Effectively the system is perturbing its task solutions slightly through randomization. The intent is to obtain probabilistically a solution that is applicable for particular instantiations of these unknown parameters.

An understanding of the way that randomization can facilitate task solutions is integral to the development of a theory of manipulation. Such a theory should try to explain the relationship between solvable tasks and repertoires of actions, with the aim of creating autonomous systems capable of existing in an uncertain world. Randomization is seen as a primitive strategy that arises naturally in the solution of manipulation tasks. Randomization is as essential to the solution of tasks as are sensing and mechanics. By formally introducing randomization into the

theory of manipulation, the thesis provides one further step towards understanding the relationship of tasks and strategies.

The thesis expands the existing framework for generating guaranteed strategies to include randomization as an additional operator. A special class of randomized strategies is considered in detail, namely the class of simple feedback loops. A simple feedback loop repeatedly considers only current sensed values in deciding on actions to execute in order to make progress towards task completion. Integral to the definition of a simple feedback loop in this thesis is the notion of a progress measure. Distance to the goal can serve as a progress measure as can some nominal plans developed under the assumption of no uncertainty. When progress is not possible the feedback loop executes a randomizing motion. The thesis shows that if the average velocity of the system relative to the progress measure points towards the goal, then the system converges to the goal rapidly. In particular, the expected time to attain the goal is bounded by the maximum progress label divided by the minimum expected velocity. A simple feedback loop in the plane is analyzed. It is shown that the rapid convergence regions of this randomized strategy are considerably better than those for a corresponding guaranteed strategy.

As part of the thesis, a simple feedback loop was implemented on a robot. The task consisted of inserting a peg into a hole using only position sensing and randomization. The implementation demonstrated the usefulness of randomization in solving a task for which sensory information was poor.

The development of randomized strategies is undertaken in the discrete and continuous domains. Most of the technical results are proved in the discrete domain, with extensions to the continuous domain indicated.

Contents

Chapter 1

Introduction

The goal of robotics is to understand physical interaction, and to use that understanding towards endowing machines with the autonomous capability of operating productively in the world. Towards realizing this goal, a large body of work has been concerned with the problem of providing robots the ability to automatically synthesize solutions to tasks specified in high-level terms. Of central importance in synthesizing these solutions is the repertoire of primitive actions that are available to a robot. It is evident that the form or even existence of a solution depends on the actions available. In turn, the actions that one is likely to consider depend strongly on one's view of the world. In recent years, the key obstacle to successfully planning and executing task solutions has been uncertainty. Uncertainty arises in a variety of forms. Often uncertainty arises from run-time errors in sensing or control. Other causes of uncertainty may be one's lack of knowledge in modelling a system or an environment. The realization that uncertainty plays a fundamental role in physical interaction has changed the character of primitive actions deemed necessary to solve particular robot tasks. For instance, in a perfect world it may be enough to specify actions of the form MOVE FROM A TO B, assuming that the path from A to B is free. In a world with uncertainty it may be impossible to guarantee the success of such an action. The work on uncertainty over the past two decades may be interpreted as searching for various primitive actions and methods of action combination that extend the class of tasks solvable in the presence of uncertainty.

The archetypical primitive action is often simply a motion in a particular direction. Sensors determine when an action should be initiated and when it should be terminated. Actions are combined by a planning or execution system whose responsibility it is to ensure that a task is completed. The outcome of a given action may be non-deterministic, as uncertainty may yield a possible range of results rather than a unique result at the termination of an action. Actions may have non-deterministic outcomes, but generally the action to be performed at a given stage in the solution of the task is deterministically fixed as a function of sensor values.

Other types of primitive actions are imaginable. For instance, instead of choosing actions deterministically as a function of sensory inputs, a system could select a motion randomly from a set of possible motions. Equivalently, a system might

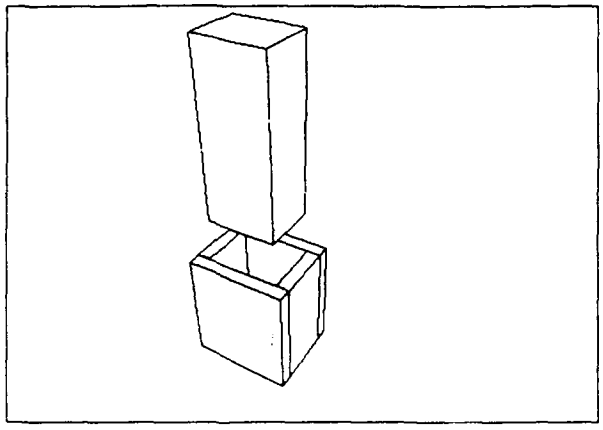


Figure 1.1: A three-dimensional peg-in-hole task.

randomly hallucinate sensor values when actual sensor values are not sufficiently precise to guide the progress of a task solution. More simply, a given action may attain a particular goal only with some non-zero probability of success but not with certainty. Nonetheless, if the action is repeatable then it makes sense to retain the action in one's repertoire. This is because one can under suitable conditions ensure eventual success by placing a loop around the action. These suitable conditions postulate the absence of trap states and lower bounds on the probability of success.

We will refer to actions in which random choices are made or in which the outcome is probabilistically determined as randomized or probabilistic actions, respectively. The purpose of this thesis is to investigate the use of randomization in the solution of robot tasks. Randomized and probabilistic actions are viewed as additional types of primitive actions whose existence is essential to the solution of many tasks.

The advantages to be gained from randomization are three-fold. First, randomization increases the class of solvable tasks beyond those solvable by bounded-step guaranteed strategies. This is because a randomized strategy need not solve a task in a specific number of steps, but must merely ensure convergence in an expected sense. Second, by tolerating local failures and circumventing these with randomization, a strategy becomes less sensitive to task details. This reduces brittleness, and, third, it simplifies the planning process.

1.1 A Peg-In-Hole Problem

Consider the task of placing a rectangular peg into a rectangular hole. See figure 1.1. One of the experiments conducted for this thesis inserted such a peg using a strategy that combined sensing and randomization. The task system consisted of a PUMA robot that manipulated the peg, and a camera system that provided position sensing.

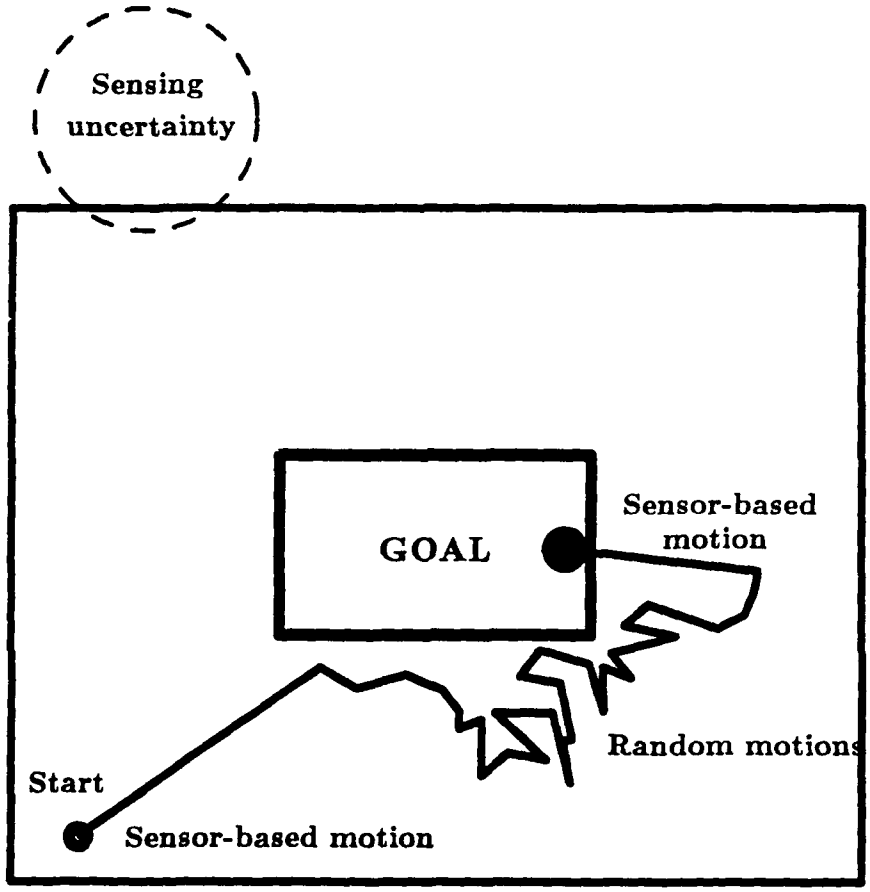


Figure 1.2: Rough sketch of the run-time character of a strategy that uses a combination of sensing and randomization to attain the goal.

Combining Sensing and Randomization

The nature of the strategy is roughly sketched in figure 1.2. The basic principle of the strategy is to make use of sensory information when possible, and otherwise to execute a randomizing motion. The purpose of the randomizing motion is to either attain the goal or move to a location from which the sensor again provides useful information. The sensing errors are represented in the figure with an error ball. For configurations of the system far away from the goal the resulting sensing information may adequately suggest an approach direction that is guaranteed to reduce the system's distance from the goal. In the figure this is indicated by a pair of long straight-line motions, one of which actually attains the goal. However, when the system is near the goal, the sensors may not be able to distinguish on which side of the goal the system is. In this case, the system will execute a randomizing motion. A possible execution trace of such motions is shown in the figure.

A Three-Degree-of-Freedom Strategy

Let us examine this strategy in more detail for the peg-in-hole problem.

The problem was restricted to a three-dimensional task, instead of the full six-dimensional problem inherent to an object with three translational and three rotational degrees of freedom. It was assumed that the peg was properly aligned vertically. This was achieved by picking up the peg from a horizontal table. However, the peg was permitted to be misaligned about the vertical axis. The translational degree of freedom corresponding to the peg's height above the hole was removed by making contact between the peg and the horizontal plate surrounding the hole. Thus the peg's remaining three degrees of freedom consisted of two translational degrees of freedom in the plane perpendicular to the vertical axis, and a rotational degree of freedom about this axis. The axis of the hole was assumed to be parallel to the vertical axis.

The system operated as follows. The camera was mounted above the assembly, looking straight down. The system would take a picture, extract edges, then try to match these to the edges of the hole and the edges of the peg. Figure 1.3 depicts an idealized picture. The hole was backlit from below by a light, so that the edges visible to the camera were primarily those bounding the open part of the hole. Having fixed on a match of image edges to the peg and the hole, the system would generate a planar motion consisting of a translation and a rotation that would roughly align the peg above the hole. Figures 1.4 through 1.6 portray some actual data obtained by the camera, along with the motion suggested by the system. The system would then try to execute this motion, and take another picture. If the picture indicated that the peg was probably above the hole and properly aligned, the system would try to insert the peg. The test for proper alignment was visibility of a pair of perpendicular edges on both the peg and the hole that were in close proximity and parallel. If the peg was not yet ready to be inserted into the hole, then the system would generate a new motion, and proceed to try again. If ever the system did not obtain useful image edges for suggesting a motion, then it would execute a randomizing motion.

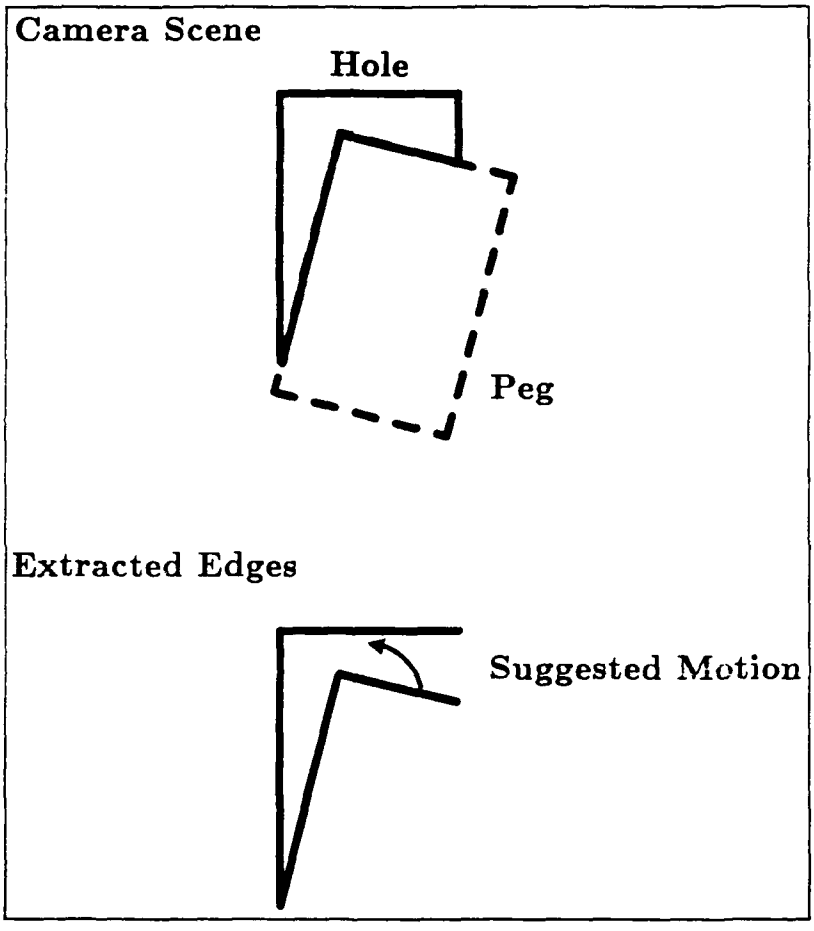


Figure 1.3: Top view of a peg-in-hole assembly. The camera extracts edges from the scene. The edges are used to suggest a motion that will align the peg over the hole.

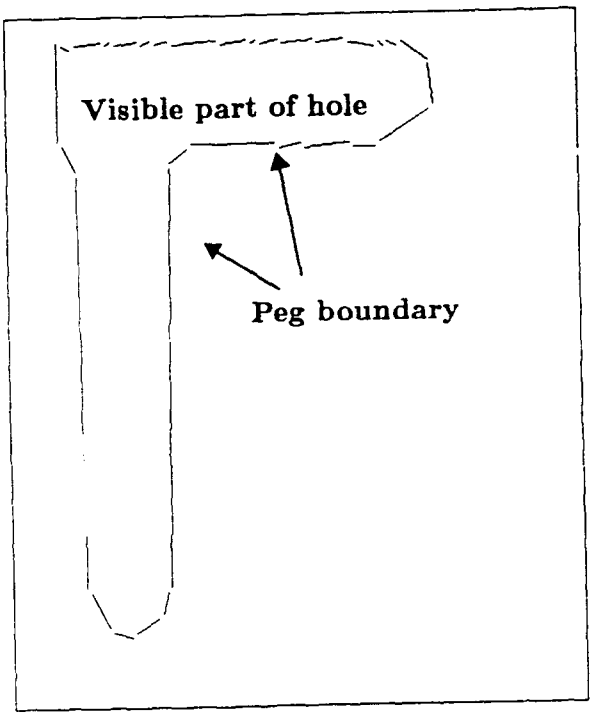


Figure 1.4: This and the next two figures show some actual image data obtained for the peg-in-hole strategy outlined in figure 1.3. The lines in this figure were obtained from an image taken by a camera looking down on the peg-in-hole assembly. The region bounded by the edges is the portion of the hole visible to the camera. The hole was illuminated from below. The lines were thus obtained by first thresholding the actual image, then looking for zero-crossings.

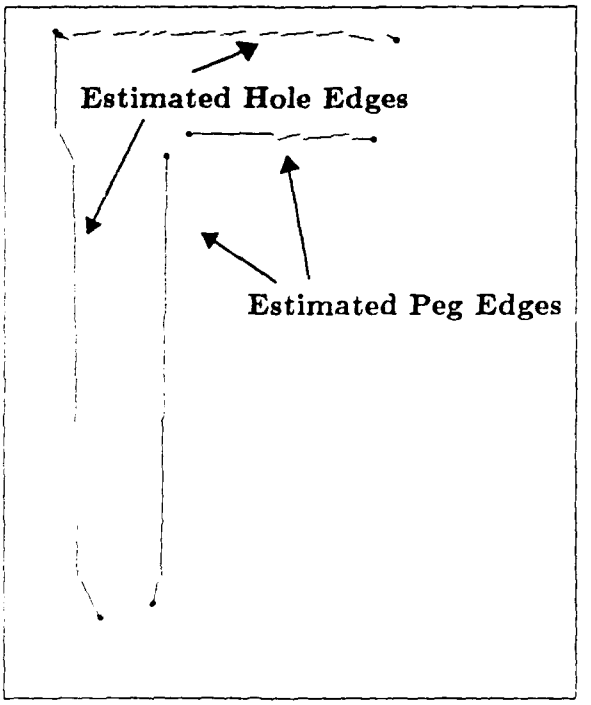


Figure 1.5: This figure shows the system's attempt to match the short image edges of figure 1.4 to the physical edges of the peg and the hole. The four vertices indicate the system's interpretation of the endpoints of the physical edges.

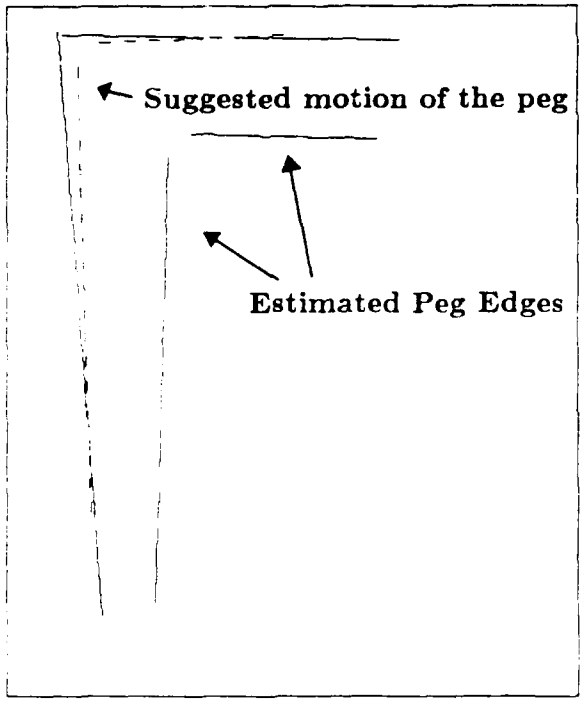


Figure 1.6: The outer two solid lines are the system's interpretation of the location of the hole boundary. The inner two solid lines are the system's interpretation of the boundary of the peg. The two dashed lines indicate the system's suggested motion. Specifically, if the peg moved precisely as suggested by the system, it would move to the location indicated by these lines.

The motion was selected in a random fashion from a collection of two-dimensional translations and rotations. In pseudo-code, the strategy was of the following form.

REPEAT until the peg is in the hole:

1. Take a picture of the assembly from above.
2. Extract zero-crossing edges from the image.
3. Try to match the image edges to the peg and the hole.
4. **IF** the edges can be matched reliably.
   THEN use these to move the peg towards the hole,
   ELSE execute a random motion.

End_repeat

Pseudo-code describing a randomized strategy for inserting a peg into a hole.

The x-y dimensions of the hole were 31.75mm × 19mm, while those of the peg were 31mm × 18mm. The material was aluminum. The random motions within the feedback loop had maximum magnitude of about 2.5mm. The insertion was started from various randomly chosen configurations within a radius of about 10mm of the center of the hole. This distance is well within the accuracy achievable using an open-loop motion of the PUMA. Indeed, the robot arm would pick up the peg several feet away from the assembly, then move it to within camera range of the assembly using a preprogrammed motion. Once within camera range, the feedback strategy outlined above would take control of the assembly.

Errors in Sensing and Control

The interesting aspect of the non-randomizing portion of this strategy is that it does not always succeed. There are two reason for this. First, the suggested motion need not be accurate, and second, the camera may not return any useful sensing information, in which case there is not even a suggested motion. The interesting failure is the second one, and it is here that randomization plays a useful role. We will return to this topic shortly.

The first type of failure arises both because of calibration errors and sensing uncertainty. Consider what it takes to transform an image motion into a robot motion. There must be some correspondence between the coordinate system of the image plane and the joint coordinates of the robot. Changing the position of the camera or refocusing can easily change this correspondence. We thus performed a rough calibration of the camera with the robot before each assembly, by executing a set of test motions, consisting of two perpendicular translations, and a rotation about a joint axis, to determine the mapping between the group of image motions and the associated joint commands. The calibration was therefore very approximate. Indeed, part of the motivation was to determine how easily one could place the peg into the

hole without requiring fine precision either in sensing or control. It is thus highly likely that the calibration contained a fixed but unknown bias. In other words, even if subsequent sensing was perfect, the initial calibration error probably introduced an unknown error into the suggested motions. Thus it would be highly unreasonable to expect the robot to insert the peg into the hole in a single motion. Additionally, there are sensing errors on each iteration. For instance, the light below the hole causes *blooming*. This means that the image edges bulge out in a curved fashion, thereby introducing error into the observed positions of the peg and the hole. In short, the non-randomizing portion of the strategy is not guaranteed to succeed in a specific predictable number of steps. Instead, the full randomized strategy operates as a simple feedback loop that eventually succeeds. This will be explained further below.

A more serious problem arises when the peg is near the hole. In this case the camera may not see any edges on either the peg or the hole, or may only see small fragments that it cannot reliably match to the peg or the hole. In part this is due to the placement of the camera. Inherently, the camera will be offset slightly to one side or the other of the assembly, and thus will not always be able to see the hole. For instance, viewing camera, peg, and hole in terms of their projections into the plane of assembly, if the peg is situated between the camera and the hole, then the camera may not be able to see any edges. Conversely, if the peg is approaching the hole from the far side of the hole relative to the camera, then the camera will likely be able to detect the defining edges of the hole and the peg throughout the approach. Thus there are preferred approach directions. Of course, the system is not aware of these, just as it is not aware of the actual biases in the calibration and sensing information.

Randomization

Now consider the state of the assembly once the peg is near the hole, supposing that the camera cannot determine any edges with which to suggest a next motion. In order to have some chance of attaining the goal, the system must make a motion. By selecting the motion randomly the system can avoid any deterministic traps that might result. For instance, if the system were to choose a motion direction deterministically, then it might have the bad fortune of moving to a location from which the sensors would direct it right back to the location at which the sensors provide no information. Thus the system would be stuck in a loop. By choosing the motion direction randomly, the system can break out of such a loop. So long as there is some chance of attaining the goal, with probabilities that are uniformly bounded away from zero, the strategy will converge eventually. Indeed, for this particular implementation we chose the maximum step size of the random motions to be on the order of 2.5mm. Thus whenever the system was within a few millimeters of the goal, it had some chance of attaining the goal upon execution of a random motion. The camera could always bring the peg to within a few millimeters of the hole. More importantly, however, the random motions permitted the system to enter a region from which the biases in the sensor-robot calibration and in the placement of the

camera actually acted in favor of goal attainment. In short, the randomizing aspect could actually ferret out approach directions from which the biases were helping rather than hindering the assembly. This is an important property of randomized strategies.

Convergence Regions

For this particular example the start configurations could be roughly grouped into four regions as indicated in figure 1.7. For one of these regions, the assembly time of the strategy was very fast, namely three motions on average. This region corresponds to the quadrant that was diagonally opposite of the camera. For the other regions the convergence times varied, although fourteen motions seems to have been a rough average (taken over fifty trials). We often observed the system finding its way into the fast region with the aid of randomizing motions, then quickly attaining the goal.

Analysis of the Strategy

Let us analyze this strategy in a very rough and approximate fashion. Suppose, for the sake of argument, that whenever the system starts in the lower right quadrant of figure 1.7, it can insert the peg in three motions on average. Experimentally, two motions were required to actually insert the peg, and one motion to recognize that the peg had been inserted. Suppose further, that if the system starts in any of the remaining three quadrants, it invariably fails to insert the peg, but instead, within two motions, places the peg above the hole in such a manner that the camera cannot extract any useful edges. Whenever this happens, the system executes a random motion, and tries again. For simplicity let us assume that the random motion moves the peg into any of the four quadrants with equal probability. Thus the probability of moving into the quadrant from which fast goal attainment is possible is $1/4$. In other words, the expected number of randomizing motions required before the system starts from the lower right quadrant is four. Since two motions are executed before each randomizing motion, the expected number of sensor-based and randomizing motions executed until the goal is attained is approximately $(2 + 1) * 4 + 3$, that is, 15.

Although this explanation is simplistic, it nonetheless provides an explanation of the observed data, as well as a description of randomized strategies in general. The important observation is that a randomized strategy is not a guaranteed strategy in the traditional sense. By a guaranteed strategy we mean a set of possibly conditional actions that are certain to accomplish a specified task in a bounded predetermined number of steps. In particular, one cannot say that a randomized strategy will succeed in a fixed predetermined number of steps. Rather, the strategy runs through a sequence of operations that merely provides some probability of success. If this sequence is repeatable and if the success probabilities sum to unity over an infinite number of trials, then one may speak of eventual convergence of the randomized strategy. Indeed, one may even be able to compute the expected number of steps until convergence. However, one cannot generally say with certainty that the strategy will succeed on any particular iteration.

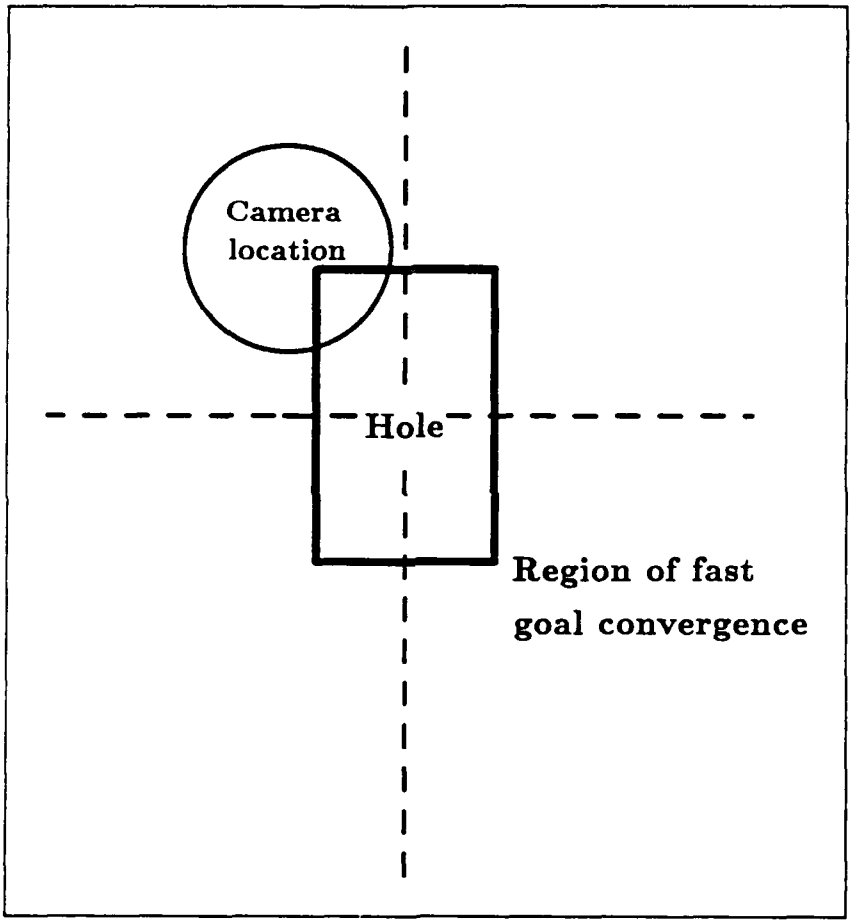


Figure 1.7: Four start regions around the hole. From one of these, the biases in the system permit fast peg insertion. From the others, the robot either attains the goal or finds its way via randomizing motions into the region from which fast goal attainment is possible.

A More General Problem

The previous analysis provides a rough explanation for the observed behavior of the feedback loop. We would like tools for analyzing and synthesizing such strategies more precisely. Most of the rest of the thesis is concerned with the development of such tools. Chapter 5 provides a detailed analysis of a simple feedback loop for attaining a circular region in the plane. This problem is an abstraction of the translational version of the peg-in-hole problem just analyzed. See figure 1.8. Recall that once the peg has made contact with the surface surrounding the hole, then the only motions required to move the peg towards the hole are translations and rotations in the plane. This is because we are assuming that the peg is aligned properly vertically. If the peg is actually cylindrical, then only translations are required. The peg was not cylindrical in our implementation. Nonetheless, the two-dimensional feedback strategy analyzed in chapter 5 provides a reasonable abstraction of the peg-in-hole problem. Higher dimensional analyses of the discussion of chapter 5 apply more generally.

We assume also that the system can recognize when the peg is directly above or in the hole. In our implementation this was usually possible because the peg would slightly drop into the hole creating a very narrow slit of light that was generally observed only when the peg was in the hole. [1]

Gaussian Errors

The simple feedback strategy will be analyzed in chapter 5 assuming Gaussian errors in sensing and control. Recall, however, that the strategy itself is formulated for more general types of errors. Similar to the implementation of the peg-in-hole example above, the feedback strategy of chapter 5 operates as a combination of sensing and randomization. Whenever the sensors provide information useful for moving towards the goal, then the strategy executes a motion guaranteed to move closer to the goal. Otherwise, the strategy executes a random motion. As we will see later, the randomization has a natural tendency to move away from the goal. In contrast, the feedback loop uses sensory information in such a way as to make progress towards the goal. However, progress is not always possible since the sensors do not always provide useful information. An important issue therefore is to determine the range of locations for which the strategy makes progress towards the goal on average. As we will prove in chapters 3 and 5, whenever the natural motion of the system is towards the goal on average, then the goal is attained quickly.

Figure 1.9 indicates the average behavior of the system for a particular set of uncertainty parameters. In particular, the sensing error is an unbiased two-dimensional Gaussian distribution with standard deviation 7/3. The qualitative shape of this graph applies more generally to different uncertainty parameters. The graph shows the expected velocity of the system as a function of the system's distance from the origin. Recall that the goal is a circle centered at the origin. A negative velocity

[1] During the course of some fifty trials there were only a couple of occasions when the system incorrectly thought that it had placed the peg in the hole.

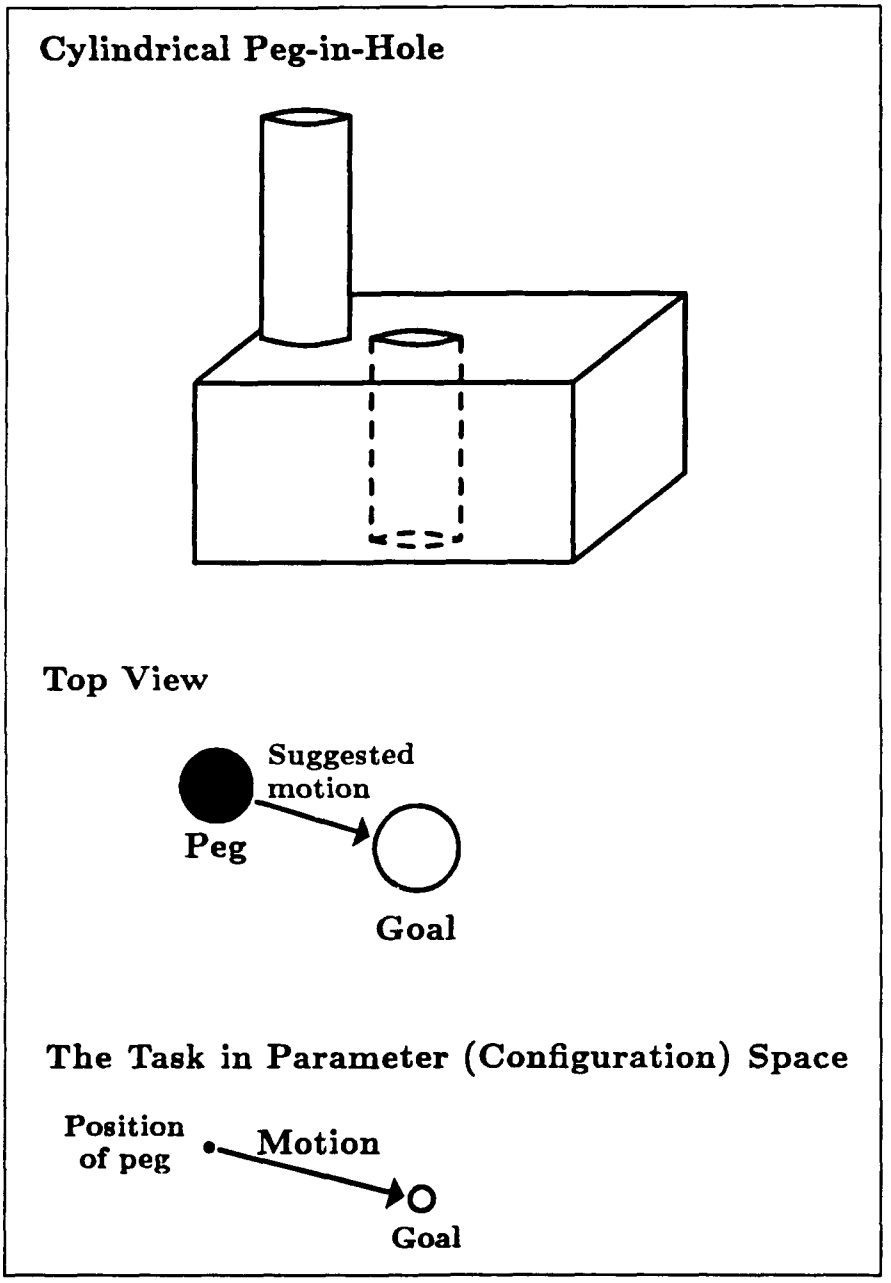


Figure 1.8: This figure shows the cylindrical version of the peg-in-hole problem of figure 1.1. The peg is assumed to be aligned vertically. Once the peg has made contact with the surface surrounding the hole, the task of moving the peg towards the hole becomes a two-dimensional problem. The task may thus be represented as the planar problem of moving a point into a circle. The point represents the position of the peg in the space whose axes are given by the two translational degrees of freedom of the peg.

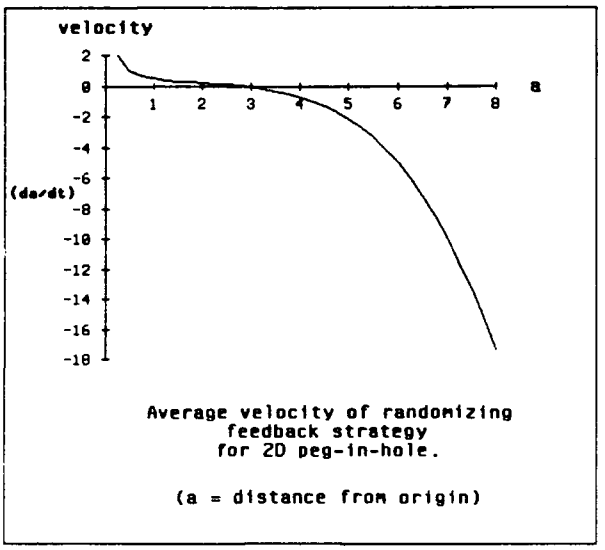


Figure 1.9: This figure shows the expected radial velocity of a simple randomized feedback strategy for the problem of moving a point into a circle, as in figure 1.8. That problem is an abstraction of the peg-in-hole problem.

The expected velocity in the figure is positive in the range $0 < a < a_0$, and negative in the range $a > a_0$, where $a_0 \approx 3$. This means that for starting positions that are closer to the origin than 3, the randomization component of the strategy naturally pushes the system away from the origin. For starting locations further away from the origin than 3, the sensing information is good enough to pull the system towards the origin on the average. This says that a goal whose radius is at least 3 would be attained very quickly.

In contrast, it turns out that a strategy which wishes to guarantee progress towards the goal on each step can do so only if the goal radius is at least 15.1. In short, the randomized strategy has considerably better convergence properties than does the guaranteed strategy.

This graph and the number 15.1 will be derived in chapter 5. The sensing error is assumed to be normally distributed with standard deviation 7/3. Similarly, the velocity error is assumed to be normally distributed with standard deviation 1/6 times the magnitude of the commanded velocity.

means that the system is making expected progress towards the origin. In particular, we see that there is a region near the origin for which the natural tendency of the system is to move away from the origin. Outside of this region the system moves towards the origin on the average. The zero-velocity point is given by approximately $a_0 = 3$ in the figure. Thus if the goal has radius bigger than a_0, the system will quickly converge to the goal. Even if the goal radius is smaller than a_0, the system will eventually converge, but now the convergence may require considerable time. Instead of drifting towards the goal on average, the system attains the goal eventually due to the diffusion character of the feedback loop. Figures 5.8 through 5.10 on pages 259–261 indicate the expected convergence times of the feedback strategy for different starting locations and different goal radii.

An important observation to take from figure 1.9 is that the randomized feedback loop has a wider convergence range than would a guaranteed strategy for attaining the goal. In order to see this, let us simply state that for the example of figure 1.9 the feedback strategy requires a sensory observation that lies at least distance 8.1 from the origin in order to guarantee progress towards the goal. Whenever a sensory observation lies closer to the goal, the feedback strategy executes a random motion. In order to **guarantee** that the only sensor values observed will be at least distance 8.1 from the origin, it turns out that the system must be at least distance 15.1 from the origin.[2] Thus, a planner wishing to guarantee, *prior* to execution time, that progress towards the goal will be made consistently at execution time would only construct plans for goals of radii larger than 15.1. On the other hand, the randomized feedback strategy converges to goals of arbitrary size. Furthermore, for the unbiased Gaussian errors used to derive figure 1.9, the strategy converges quickly for goals of radii as small as 3. This is because the expected approach velocity points towards the goal whenever the system is at least distance 3 from the origin.

1.2 Further Examples

1.2.1 Threading a needle

There are numerous examples of manipulation tasks in which randomization arises naturally. For instance, consider the task of threading a needle. Without perfect control and perfect sensing, it is unlikely that one can thread a needle on a specific try. Nonetheless, within a reasonable starting location near the eye of the needle, there is a definite chance of success on each attempt to insert the thread, so that success can be guaranteed by trying repeatedly. This is an example of a probabilistic action around which a loop has been placed.

[2]These numbers are derived form a particular sensing model that will be explained in more detail in the rest of the thesis. See in particular sections 2.2.3 and 5.2.

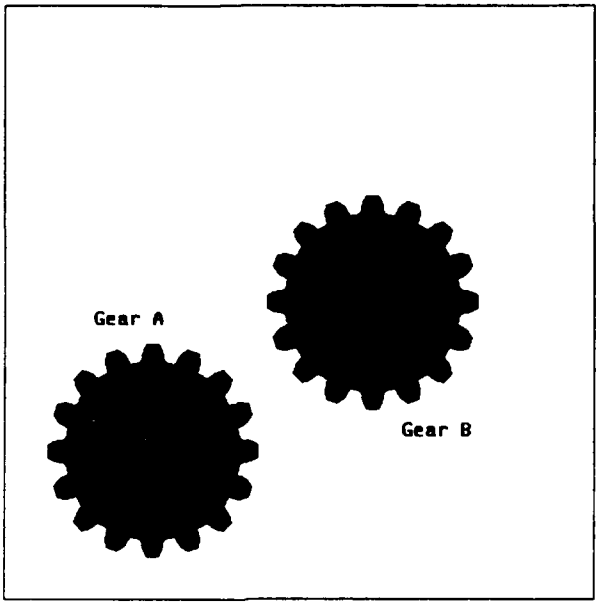


Figure 1.10: Two gears.

1.2.2 Inserting a key

Similar examples are given by tasks such as inserting a key into a lock or closing a desk drawer that is jamming. In the key–lock task the solution consists of moving the key near the keyhole, then moving the key back and forth if necessary while reducing the distance to the hole, until the key actually slides into the keyhole. Once the key is in the lock, one may have to jiggle it back and forth while pushing in order to fully insert the key. The example of closing a desk drawer is similar to this last step. If the drawer jams, one may randomly jiggle it while pushing inward, to overcome any jamming forces.

1.2.3 Meshing two gears

A wonderful example is given by the task of meshing two gears (see figure 1.10). Donald ([Don87b] and [Don89]) first used this example to demonstrate a task in which solutions cannot be guaranteed but for which there is some hope of success. His thesis was that a robot should attempt to solve such tasks, so long as at the end of each attempt the robot is able to distinguish between success and failure. For the gear-meshing case, should the success not be directly visible, Donald suggested a test that consists of trying to rotate one gear. If the other gear rotates as well, with the proper gearing ratio, then the meshing operation is known to have completed successfully. Otherwise, it has failed. In the context of the randomized actions of this thesis, the attempt to mesh the gears will play the role of a non-deterministic action, around which we will place a loop whose active randomization guarantees eventual success.

In order to get a flavor of the approach, consider a simplified version of the gear-meshing problem in which one can move the gears towards each other perfectly, so that the centers of rotation travel on the straight line joining them (this might be possible if the gears are mounted on a telescoping device constraining their centers of rotation). As the gears are brought near each other, they will mesh if they are properly aligned. In other words, for some set of starting orientations, the two gears will mesh if brought together. The range of starting orientations that permit successful meshing is some subset of the two-dimensional space $[0, 2\pi] \times [0, 2\pi]$ that describes the possible orientations of the two gears. Suppose that one cannot sense or control the orientation of the gears well enough to be able to ensure that the gears are properly oriented. If initially the gears are randomly oriented, then the ratio of the area of the successful starting range to $4\pi^2$ is the probability of success on any given try. A randomized strategy for meshing the gears consists of first spinning the gears to achieve a random orientation, then bringing them together in an attempt to mesh them, followed by a test to determine whether they have indeed meshed properly. This action is repeated until the test indicates that the gears have been meshed. The expected number of attempts until success is simply one over the probability of success on a given try.

This example raises a number of important issues. First, let us consider the probability of success on a given iteration. In order to specify the strategy of looping around a primitive action one really does not need to know what this probability of success is. It is sufficient to know that on each try there is some chance of success and that the sum of the probabilities of success over an infinite number of trials is one. For instance, it is sufficient to know that the probability of success on each trial is larger than some non-zero constant.

While the specification of the strategy does not depend on the probability of success, it is nonetheless sometimes desirable to compute this probability, either to ascertain that it is non-zero or to compare it with other possible strategies. This entails computing the area of the range of initial orientations that permit successful gear meshing. Figure 1.11 portrays this range in a highly approximate fashion. Essentially the range of successful initial orientations consists of a set of diagonal stripes in the space of orientations of the two gears. The number of stripes depends on the number of gear teeth, and the inclination of the stripes depends on the gearing-ratio. The center axes of the stripes correspond to orientations of the two gears at which the gears are perfectly meshed. The stripes themselves include orientations at which the gears are not perfectly meshed, but from which the gears will compliantly rotate to perfect meshing if they are pushed together. Computing the exact shape of these stripes is in general a difficult task, which depends on the exact geometry of the parts and on the coefficient of friction between them. The basic idea is to start from a goal consisting of those orientations at which the gears are perfectly meshed, then backchain, recursively determining all those points that can move compliantly toward the goal under a given applied force. The problem is complicated by the rotational compliance of the gears. This backchaining process is known as computing *preimages* or *backprojections* (see [LMT], [Mas84] and [Erd84]). We will refer to this approach as the **LMT** preimage planning approach. Donald (see [Don87b] and

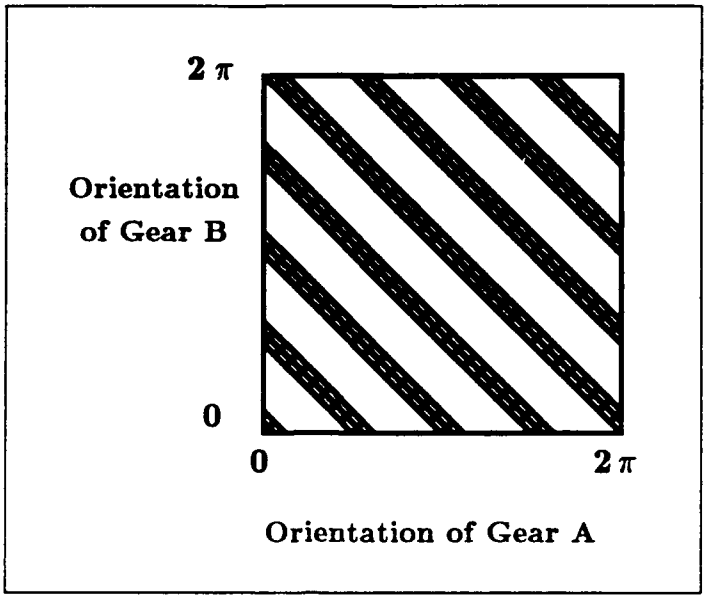


Figure 1.11: Schematic representation of the range of initial orientations that permit successful gear-meshing. These are indicated by the hatched areas. [The figure corresponds to gears with only four teeth. Realistic gears generate more stripes.]

[Don89]) has investigated approximate techniques for computing such backprojections in the gearing case. We will not examine those techniques here. Instead, we will convey the basic idea of how one might compute success probabilities with a slightly simpler example.

If we fix the orientation of one of the gears, the successful starting orientations of the other gear form a periodic pattern of disconnected intervals. This pattern looks very much like a sieve, and indeed we can think of the gear as a sieve that filters out bad orientations of the other gear or, more generally, improperly shaped gears. Let us look then at a sieve to demonstrate in a simpler setting the ideas behind randomized strategies.

Figure 1.12 shows a simple grating that acts like a one-dimensional sieve, permitting some two-dimensional objects through but not others. Let us suppose that the object we would like to get through the sieve is a square, as shown in figure 1.13. Assume the object can only translate, not rotate. Relative to the indicated reference point on the object, the translational constraints imposed by the sieve on the object are as shown in figure 1.14. This is the configuration space (see [Loz81] and [Loz83]) representation of the sieve. Moving the object through the real sieve corresponds directly to moving a point through this configuration space sieve. The representation depends of course on the object being moved. Indeed for objects that are too large, the configuration space sieve is simply a solid horizontal slab, indicating

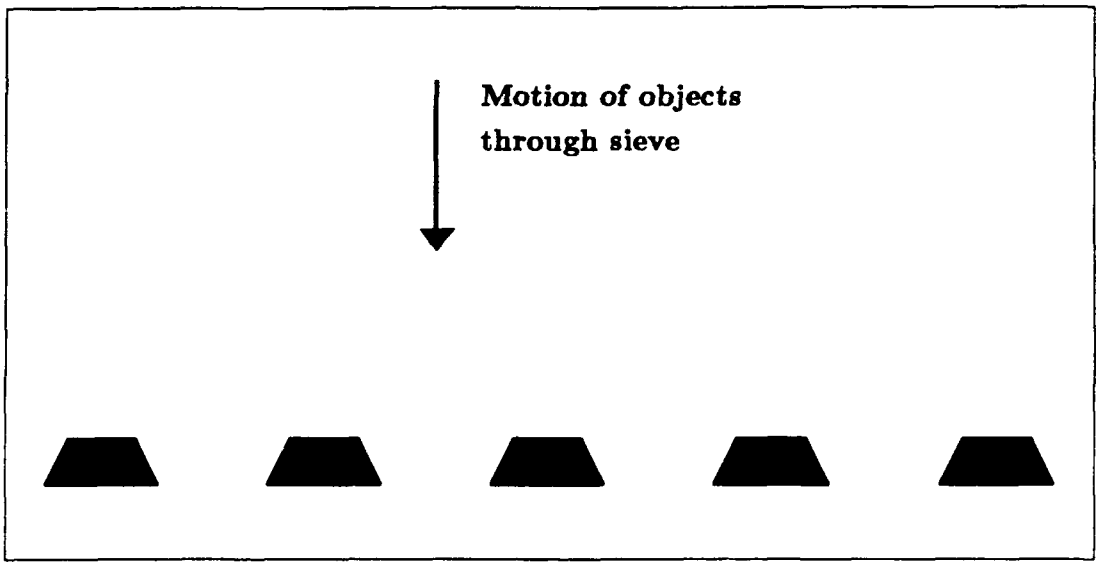


Figure 1.12: A grating of two-dimensional obstacles. The grating acts as a one-dimensional sieve, only allowing objects small enough to move through the sieve.

that the object cannot be moved through the sieve.

Given the configuration space sieve, we are now in a position similar to the gear-meshing example. In particular, let us suppose that the analogue to moving the gears together consists of translating the object vertically downward (for instance, under the influence of gravity). Then, for certain starting configurations, the object will translate through the sieve, while for other configurations it will become stuck on the sieve elements. Thus the sieve also acts as a configuration sieve, filtering out certain initial starting configurations of the object. Of course, that is not exactly the purpose of the sieve. After all, one would like the object to translate through the sieve. In order to ensure this, one shakes the sieve, or equivalently, one randomizes the initial position of the part. This operation corresponds to the act of twirling the gears in order to randomize their configurations on each meshing attempt.

Let us compute the probability of success for the sieve example. First, let us assume that there is no control uncertainty, so that the object translates straight down when commanded to do so. Figure 1.15 portrays those start configurations from which the object is guaranteed to pass through the sieve when translating downward (recall that the part is represented by a point in its configuration space). Suppose that the sieve is periodic and unbounded, and suppose further that the start configuration of the object is uniformly distributed. One then sees that the probability of success is simply the ratio of the length of a hole in an elemental period of the sieve to the full length of the elemental period.

In the previous computation, it was enough to look at one-dimensional quantities

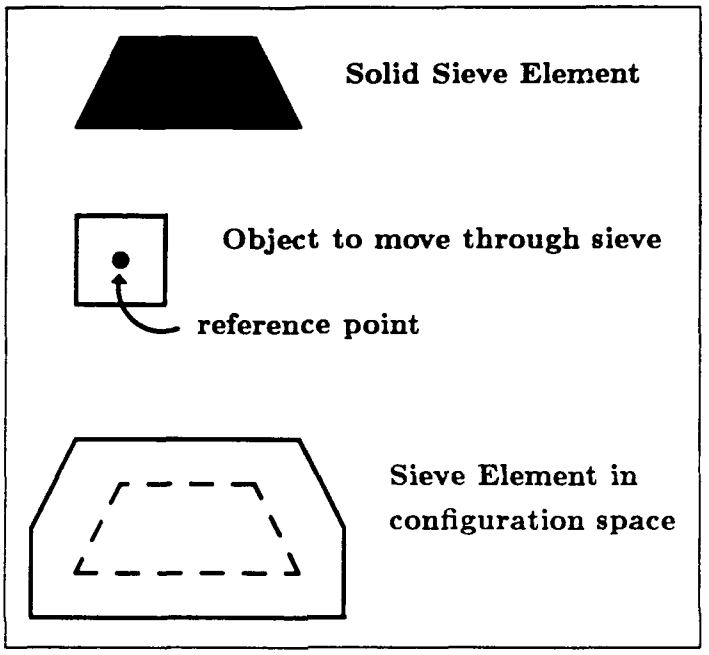


Figure 1.13: This figure shows the constraints imposed on the motion of the square by the trapezoidal sieve element. The bottom polygon describes the locations of the reference point of the square for which there would be contact between the square and the trapezoid.

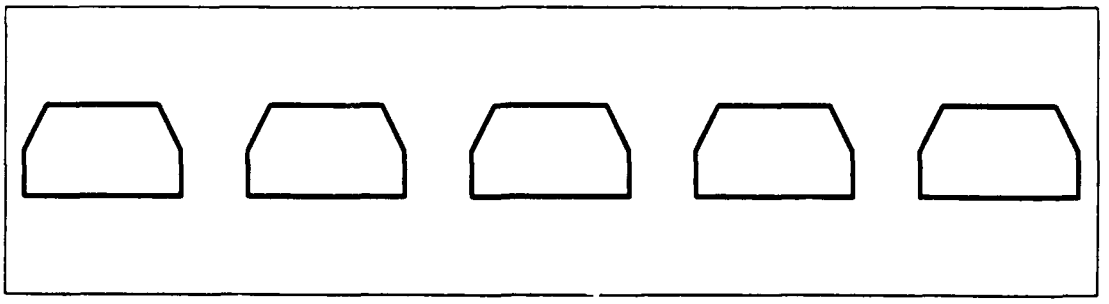


Figure 1.14: This figure shows the configuration space sieve corresponding to the real space sieve of figure 1.12 for the motion of a square as in figure 1.13.

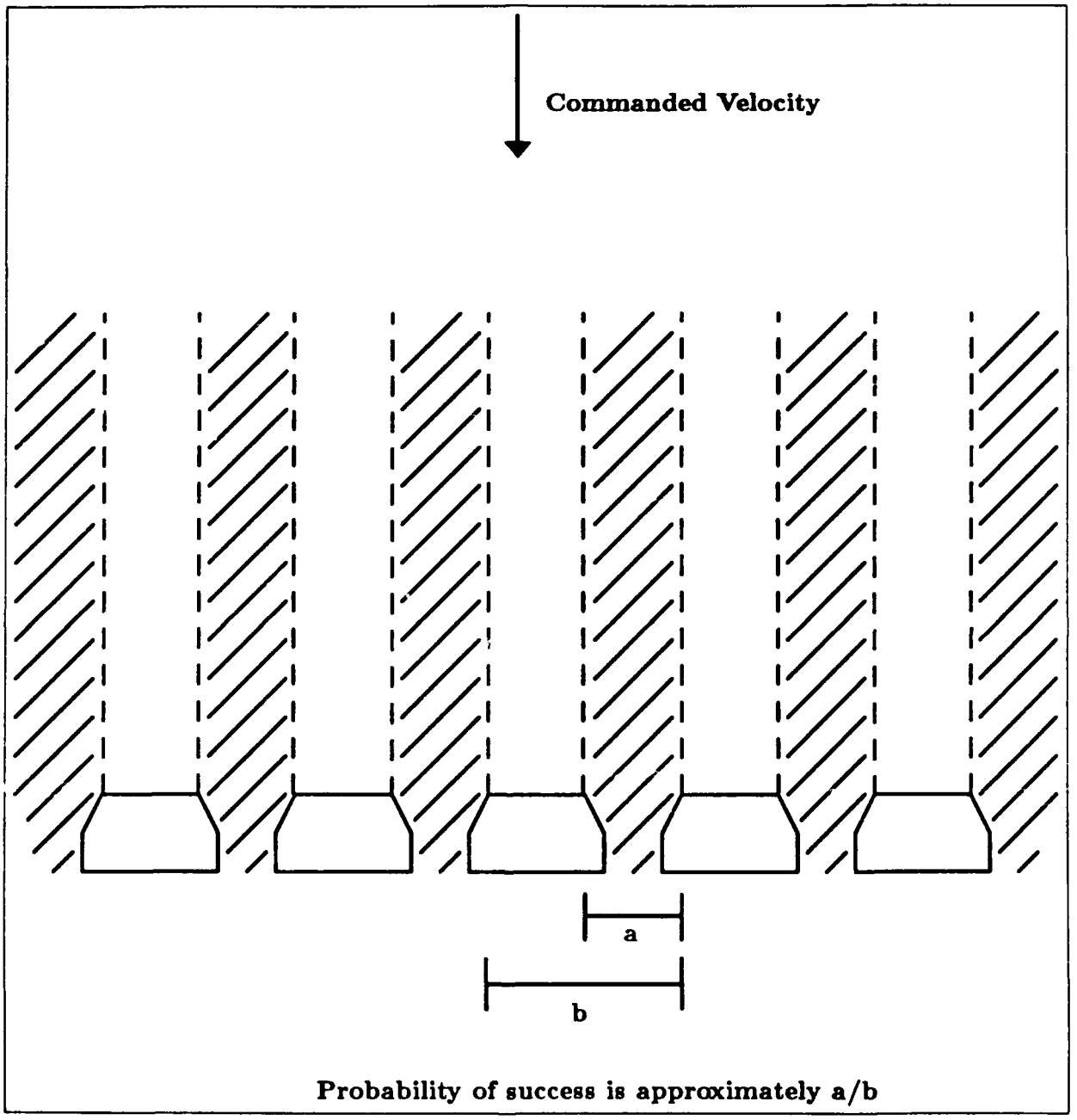


Figure 1.15: Perfect velocity preimage. For starting locations in the shaded area, the system is guaranteed to pass though the sieve by moving straight down. For other locations, the system will get stuck on a horizontal edge. If the starting location is uniformly distributed, then the probability of passing through the sieve is approximately a/b.

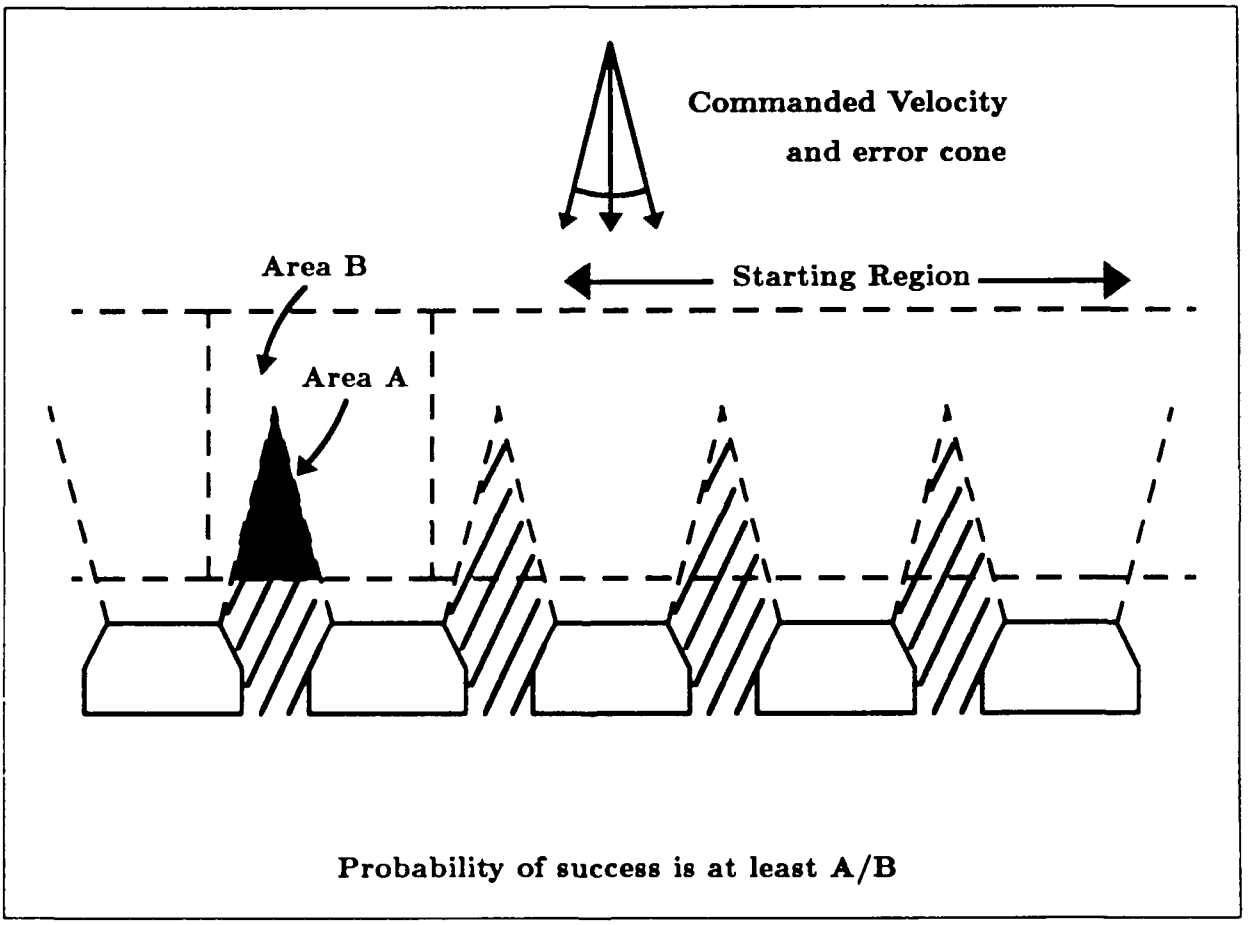


Figure 1.16: Preimage assuming velocity error. For starting locations in the shaded area, the system is guaranteed to pass though the sieve, given the velocity error cone. For other locations, the system may get stuck on a horizontal edge. If the starting location is uniformly distributed in the infinite horizontal strip, then the probability of passing through the sieve is at least A/B.

in computing the probability of success, since the vertical coordinate of a point above the sieve did not matter in determining whether the point would translate through the sieve. However in general one needs to compute the ratio of the area of successful starting configurations to the area of possible starting configurations. Suppose, for instance, that whenever a translation is commanded the actual motion lies within some velocity error cone about the nominal commanded velocity. Then the set of initial configurations from which translation through the sieve is guaranteed changes. Indeed, the successful starting configurations are delineated by triangles that are determined by erecting the velocity error cone above the sieve holes, as shown in figure 1.16. Suppose that the initial configuration of the object is known to lie in some region, uniformly distributed. Consider those portions of the triangles that lie within this starting region, and sum up the areas of these portions. Then the probability of success is given by the ratio of this area to the full area of the starting region. This computation is also indicated in the figure for a periodic sieve with a periodic starting region.

Actually, the probability thus computed is an underestimate. This is because the probability is determined only by considering configurations from which passage through the sieve is guaranteed, independent of the actual motion taken within the velocity error cone. Such regions are known as *strong preimages* (see [LMT]). It is, however, also possible that some points that lie outside of these strong preimages may for some possible error velocity pass through the sieve. However, since this passage cannot be guaranteed, without further information, one cannot say anything about how the possibility of success for these start configurations affects the total probability of success. If the probability distribution of the velocity errors is known, then it can be used to compute an additional term that figures into the probability of success. Without such knowledge, however, we can imagine that no point outside of the strong preimages ever passes through the sieve, and thus our original probability computation is the best possible lower bound.

Another issue raised by these examples concerns the need for randomization. We will discuss this issue further in the next section, but let us briefly consider the question of randomization in the context of the gear and sieve examples. One might wonder why it is ever necessary to randomize the start configuration of a part, as opposed to deterministically searching the set of possible start locations. For instance, in the gear example, even if the orientations of the gears are not measurable well enough to ensure proper initial alignment, one could imagine rotating one or both of the gears slightly after each meshing attempt, then retrying. If the rotation is small enough, then this process should eventually encounter a starting orientation from which successful meshing is possible. Unfortunately, there are some problems with this approach. First, it may be impossible to rotate the gears finely enough to guarantee that the rotation will not just jump over the successful start orientation. And second, after a failed meshing attempt the configuration of the gears will have changed, so that it is not at all clear that incremental rotations will eventually encounter a successful starting configuration. In principle, the system could get into a loop, starting from a given unsuccessful orientation, rotating during the failed attempt to an orientation

exactly offset from the start orientation by the angle of increment, thus ensuring that the new start orientation after incrementing is again the old start orientation, and so forth. Of course, if one's predictive capabilities are good enough, one could detect the potential for such a loop, but that is not always the case. A straightforward method of avoiding the possibility of a deterministic loop is to randomize the initial conditions. We will discuss this approach in more detail in the next section.

There is another reason for randomizing, which again relates to the accuracy with which one can model the world. In the sieve example, for instance, the spacing between sieve elements may be slightly non-uniform, so that one cannot predict exactly where a hole will be. Taken over a large segment of the sieve, the density of holes to non-holes may be the same as in the uniform case, but it may vary locally. Thus it may make sense to randomize the start location to take advantage of the high overall probability of success, avoiding possibly low local probabilities of success. Let us make this argument more precise. Suppose that in a perfectly shaped sieve, the period of the sieve has length b, of which length a is free space, and length $b-a$ is an obstacle. Thus the probability of success (assuming perfect control) is a/b. See again figure 1.15. Now suppose that the sieve is not built very well. Instead there are two types of sieve sections. In Type One sections the hole has size $a+\epsilon$, while in Type Two sections the hole has size $a-\epsilon$, where ϵ is some positive number satisfying $0 < \epsilon < \min\{a, b-a\}$. Suppose that the underlying period of the sieve still has size b, and that the two types of sieve sections occur with equal frequency when viewed over the entire sieve, although locally one or other type may dominate. If the state of the system happens to be in a region in which there are only Type One sieve sections, then the probability of success is $(a+\epsilon)/b$, whereas if the system happens to be in a region in which there are only Type Two sections, then the probability of success is $(a-\epsilon)/b$. If ϵ is close to a, then the probability of success might be very near zero if the system is in this second region. However, if the system first randomizes its initial position, so that it starts off with a uniformly chosen initial position, then the resulting probability of success is given again by a/b.

We will discuss a related example in section 2.4. Another related example is given by a person trying to open a door in the dark. Suppose he has n keys of which k will open the door. If he tries the keys in order, in the worst case he may need to try $n-k+1$ keys before success, but if he tries them randomly (with replacement), then, although the worst case is now unbounded, the expected number is n/k. If n and k are large and k is comparable to n, but still considerably less than n, then it makes sense to try the randomized approach. This is essentially the motivation behind the use of probabilistic algorithms in computer science. If k is small, then the deterministic approach is preferable. However, even in this case, if the deterministic approach is subject to failure, in that the person may drop the keys or forget which keys he has already tried, then the randomized approach is again useful.

To summarize, randomization is useful in two ways. First, randomization foils an adversarial world that might cause a deterministic search to loop. Second, randomization may compensate for imperfect world knowledge, by ensuring that successful actions are taken at least occasionally, and in some cases by ensuring that

successful actions are taken with high enough frequency.

1.3 Why Randomization?

The main purpose of randomization is to increase the class of solvable tasks. In particular, randomized strategies are useful for solving tasks for which there is no guaranteed solution but for which there is some probability of success. A guaranteed solution in this context refers to a strategy consisting of a set of possibly conditional actions that are certain to accomplish a task in a bounded predetermined amount of time. In contrast, a randomized strategy is expected only to attain the goal in some expected amount of time. While giving up predetermined convergence, randomized strategies provide a tool for solving a broader class of tasks.

Randomization also increases the class of solvable tasks by reducing the demands made on modelling and prediction. For instance, in the peg-in-hole task at the beginning of this chapter, we were not required to model very accurately the errors introduced by the calibration process. More generally, one can imagine tasks in which geometrical errors in the modelling of parts prevent guaranteed solutions. For instance, there might exist slight nicks and bumps on the hole surface, which could prevent successful entry of the peg into the hole. In general, it is very difficult to plan explicitly for such irregularities. However, for a large class of such irregularities the system can avoid becoming permanently stuck by wiggling the peg slightly, that is, by introducing randomized motions.

Reducing the demands on modelling and prediction also permits simpler solutions to tasks for which there might actually exist guaranteed strategies. In addition, reducing the knowledge requirements of a strategy reduces its brittleness.

One question remains. It deals with the difference between active randomization and probabilistic or non-deterministic actions. In the context of this thesis, to say that a strategy or an action is *probabilistic* is to say that it has some non-zero probability of success, but may not be guaranteed to succeed. More formally, an action is probabilistic if its effect on each state is modelled as a set of configurations, each of which has some non-zero probability of occurring. Often a probabilistic strategy will consist of some loop around a probabilistic action, the purpose of the loop being to guarantee eventual convergence.

More generally, a strategy or action is said to be *non-deterministic* if its outcome is modelled as a set of possible configurations. The non-deterministic model is intended as a worst-case model. It says simply that an action might cause a transition to any one of a set of possible configurations. However, nothing is said about the actual likelihood of that transition occurring.

While an action may be probabilistic, the decision to execute that action is often deterministic. In other words, given certain sensor values, the system selects a certain action in a completely deterministic fashion. It is simply the outcome of the action that is probabilistic or non-deterministic. An alternate approach is for a system to actively make random choices in selecting actions. This process is what we have been

calling *randomization.*

We have already indicated in the sieving example the usefulness of randomization. However, it may not be clear why randomization is ever really required. After all, one could imagine that a system could simulate in a deterministic fashion a randomizing system, simply by enumerating in some order all possible random decisions of the randomizing system, until the goal is attained.

One possible benefit of active randomization might be improved convergence times. There are certainly arguments from the theory of randomized algorithms that suggest that randomization can speed up convergence of certain tasks. Indeed, we will exhibit an example in chapter 3 for which randomization does speed up convergence. However, the problem here is slightly different, in essentially three ways. First, unlike decision problems in algorithms, when moving in the physical world one cannot arbitrarily restart the problem to improve convergence. For instance, for decision problems in Bounded Probabilistic Polynomial time, one can repeatedly ask the decision question, thereby making the probability of error as small as desired. Furthermore, this may be done in a polynomial amount of time. In contrast, once a robot has moved, it may have introduced uncertainty into its configuration, and thus may not be able to restart from the same location should it fail to attain its goal. To some extent one can define this issue away, by insisting that it be possible to place a loop around any probabilistic sequence of actions. However, the basic difference remains. Second, many robot planning problems in the presence of uncertainty are at least PSPACE-hard (see [Nat88], [CR], and [Can88]). Thus the hope for polynomial speedup by moving to probabilistic algorithms seems futile in general (see [Gill]). Third, our main interest lies in extending the class of solvable tasks, with performance issues entering as a secondary motivation. Thus the question of whether randomization is ever required enters at the level of task solvability rather than purely at the level of convergence time.

The need for randomization arises in the context of non-deterministic actions. When actions are probabilistic, one can, at least in principle, compare different decisions based on their probability of success, then select that decision which maximizes the probability of success. No randomization is required. However, in the setting of non-deterministic actions, one must be prepared to handle worst-case scenarios. This means that one should view uncertainty as an adversary who is trying to foil the system's strategy for attaining the goal, and who will therefore always choose that outcome of a non-deterministic action that prevents the system from attaining its goal. Again, it may seem that one can enumerate all decisions and actions, then select that sequence of actions that is guaranteed to attain the goal despite the most devilish adversary. Indeed, this is the approach taken in planning systems that generate guaranteed plans, that is, plans guaranteed to attain the goal in a predetermined bounded number of steps. However, not all tasks admit to guaranteed solutions. The interest of this thesis is in tasks for which there may not exist any guaranteed plan, or for which finding a guaranteed plan may be very difficult. In that setting randomization can play a useful role, in that it can prevent an adversary from forev foiling the goal-attaining strategy. We should note that there is a tacit

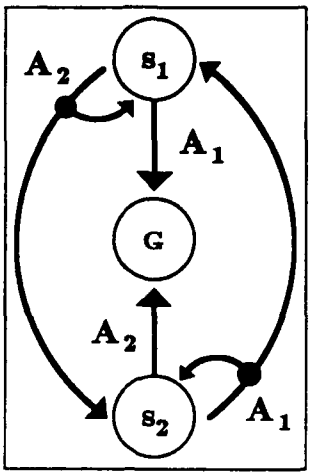


Figure 1.17: This is a state graph with non-deterministic actions. There is no guaranteed strategy for attaining the goal if the state of the system is unknown. However, by randomly and repeatedly executing one of the actions A_1 or A_2, the goal is attained in two steps on average.

assumption here that nature, that is, the adversary, cannot control or observe the dice used to make the randomizing decisions. We now demonstrate the usefulness of randomization with a very simple example.

Imagine a discrete three-state system, as shown in figure 1.17. There is one goal state G, and two other states labelled as state s_1 and state s_2. Additionally, there are two actions, A_1 and A_2, that have non-deterministic outcomes. If the system is in state s_1 then action A_1 is guaranteed to move the system to the goal. However, action A_2 will non-deterministically move the system from s_1 either back to s_1 or to the other state s_2. Similarly, if the system is in state s_2, then action A_2 is guaranteed to attain the goal, while action A_1 will non-deterministically either remain in s_2 or move to state s_1. Suppose that the only sensing available is goal recognition. In other words, the system can detect goal attainment, but cannot decide whether it is in state s_1 or s_2. We observe that there is no guaranteed strategy for attaining the goal. For any deterministic sequence of actions there is some interpretation of the diagram for which the sequence fails to achieve the goal. Said differently, from a worst-case point of view, no finite or infinite deterministic strategy is guaranteed to attain the goal.

As an example, consider the sequence of actions $A_1; A_1; A_2; A_1; A_2; A_2$. The following is a possible sequence of transitions that fails to attain the goal.

$$s_2 \xrightarrow{A_1} s_2 \xrightarrow{A_1} s_1 \xrightarrow{A_2} s_2 \xrightarrow{A_1} s_1 \xrightarrow{A_2} s_1 \xrightarrow{A_2} < \text{whatever} > .$$

In order to prove that there is no guaranteed strategy for attaining the goal, imagine an adversary who can look ahead to the next action A_i, and uses the current

action to either stay in the current state or move to the other non-goal state. In particular, the adversary can always move to state s_j, with $j \neq i$, where the index i is determined by the action A_i. (Here both i and j are either 1 or 2.)

The introduction of an adversary is just a proof artifice of course. There is no need to actually have someone look at a purported strategy for attaining the goal. The point is that even without an adversary, the transition diagram might behave as if there were such an adversary, for any **fixed** deterministic strategy. For instance, the transition diagram might be the visible portion of a considerably more complex machine or natural process, whose transitions govern the apparently non-deterministic transitions of A_1 and A_2.

One question one might ask is how complex such a hidden state diagram must be to foil a particular deterministic strategy. In particular, if one limits the complexity of the hidden diagram, then sufficiently long deterministic strategies will eventually attain the goal. We will not delve into this question.

Another question concerns the importance of the term "**fixed**". If one varies the strategy then one increases the likelihood of obtaining a guaranteed strategy. Of course, varying a deterministic strategy in a deterministic manner yields another deterministic strategy. Instead, suppose that one varies the strategy by randomizing. Then we see that there exists a randomized strategy whose expected convergence time is very low, namely two steps. This strategy randomly chooses between actions A_1 and A_2 on each step, choosing each action with probability 1/2. Since the system is in some state s_i, the strategy will choose the correct action A_i for that state with probability 1/2. This is true independent of the behavior of the system. Thus, by a waiting time argument, the expected time until the system guesses the correct action is two. In turn this says that the expected time until the goal is attained is no greater than two. [It may actually be less than two, if the underlying transitions are themselves probabilistic rather than adversarial.] This example shows clearly how randomization can solve tasks for which there are no guaranteed strategies, and for which no deterministic simulation of the randomization is guaranteed to solve the task.

The argument of the example above is essentially a worst-case versus expected-case analysis. It may seem strange to compare worst and expected cases. However, there are two important observations to take from this example. First, there is a major advantage to be gained by considering the expected case rather than the worst case. This is because the task of attaining the goal is solvable **only** in the expected case, not in the worst case. Second, the expected case convergence time is computed over randomizing decisions actively made by the run-time system, not over externally defined probability distributions. In particular, the system has control over this expectation on any attempt to complete the task. It is not an expectation computed over different possible world models of the actions A_1 and A_2. Rather the upper bound on the expectation applies for every possible interpretation of the underlying non-deterministic model.

As a final comment, let us observe that often probabilistic actions may have the same effect as active randomization on the system's part. For instance, if the non-

deterministic transitions of A_1 and A_2 were probabilistic, with transition probabilities 1/2, then the system could simply execute action A_1 repeatedly. No randomization would be required, since the physics of the problem would effectively provide the required randomization. If the system originally started from state s_1, the strategy would succeed in a single step, whereas if the system started from state s_2, the strategy would succeed in the expected time of three steps.

1.4 Previous Work

Work on planning in the presence of uncertainty goes back in time as far as one can imagine. Credit for the modern approach probably goes to Richard Bellman [Bell], who formulated the dynamic programming approach that underlies much of optimal control and decision theory. His ideas were themselves based to some extent on the calculus of variations and game theory. See [Bert] for an introduction to dynamic programming in the discrete domain, and see [Stengel] for an overview of techniques in optimal control.

1.4.1 Uncertainty

Within the domain of robotics, uncertainty has always been a central problem. Much of the work on compliant motion planning was motivated by a desire to compensate for uncertainty in control and inaccuracies in the modelling of parts. The aim was to take advantage of surface constraints to guide assembly operations. Inoue [Inoue] used force feedback to perform peg-in-hole assembly operations at tolerances below the inherent positional accuracy of his manipulator. Simunovic (see [Sim75] and [Sim79]) considered both Kalman filtering techniques in position sensing and the use of force information to guide assembly operations in the presence of uncertainty. In conjunction with this work there grew an interest in friction and the modelling of contact to describe the possible conditions under which an assembly could be accomplished successfully. See [NWD], [Drake], [OHR], [OR] and [Whit82]. More recent work with an emphasis on understanding three-dimensional peg-in-hole assemblies in the presence of friction and uncertainty includes [Caine] and [Sturges].

1.4.2 Compliance

The formalization and understanding of compliant motion techniques received several major boosts. Whitney [Whit77] introduced the notion of a generalized damper as a way of simplifying the apparent behavior of a system at the task level. The generalized damper is a first-order description of a system. A zeroth-order description is given by a generalized spring. In this direction, Salisbury's [Sal] work on generalized springs provided a means of stiffness control for six degrees of freedom. Several researchers considered a form of control known as hybrid control (see the article [Mas82b] for a pointer to these various researchers, and more generally the book

[BHJLM]). The work of Mason [Mas81] contributed to the understanding of compliant motions by modelling and analyzing compliance in configuration space. In particular, he introduced and formalized the ideas of hybrid control, showing how these could be modelled naturally on surfaces in configuration space. The basic approach is to maintain contact with an irregular and possibly unknown surface, by establishing a force of contact normal to the surface, while position-controlling directions tangential to the surface of contact. In short, uncertainty is overcome in some dimensions. Raibert and Craig [RC] describe a combination of position and force control in their implementation of a hybrid control system. See also [Inoue] and [PS] for earlier work on hybrid control.

1.4.3 Configuration Space and Motion Planning

The notion of *configuration space* was introduced into robotics by Lozano-Pérez (see [Loz81] and [Loz83]), as a means of characterizing a robot's degrees of freedom and the constraints imposed on those degrees of freedom by objects in the world. A point in configuration space corresponds to a configuration of the robot in real space. Thus configuration space is a means of transforming a complicated motion planning problem into the problem of planning the motion of a point in a (possibly) higher-dimensional space whose axes are the robot's degrees of freedom. The roots of these ideas may be found in [Udupa], who transformed the problem of moving a robot among a set of obstacles into the problem of moving a point among a set of transformed obstacles. See also [Loz76], who used configuration space in the context of grasping parts. The motivation for configuration space was initially to solve the obstacle avoidance problem. In particular, the configuration space of an object provides a geometric description of the set of collision-free configurations of the object, and thus the basis for planning algorithms. Much work has occurred in obstacle avoidance since then; see below for a partial list.

An important observation made by Mason's paper [Mas81] is that configuration space possesses dynamic properties as well as purely kinematic properties. Thus the normals to configuration space surfaces have dynamic significance. In particular, one can push on a configuration space surface and experience a reaction force. This observation meant that hybrid control could be viewed nicely in configuration space. Additionally, the dynamic information of configuration space was later used by Erdmann [Erd84] to model friction in configuration space.

As we have indicated, much of the geometric work on motion planning provided a foundation for the subsequent and parallel work on planning with uncertainty. Investigation of the motion planning problem finds its roots in the works of Brooks [Brooks83]; Lozano-Pérez and Wesley [LPW]; Reif [Reif]; Schwartz and Sharir [ScShII] and [ScShIII]; and Udupa [Udupa]. For further foundational work in the area, both for a single robot and for several moving robots, see Brooks and Lozano-Pérez [BLP]; Lozano-Pérez [Loz86]; Canny [Can88]; Canny and Donald [CD]; Donald ([Don84] and [Don87a]); Erdmann and Lozano-Peŕez [ELP]; Fortune, Wilfong, and Yap [FWY]; Kant and Zucker [KZ]; Khatib [Khatib]; Kodits hek [Kodit]; Hopcroft,

Joseph, and Whitesides [HJW]; Hopcroft, Schwartz, and Sharir [HSS]; Hopcroft and Wilfong ([HW84] and [HW86]); O'Dunlaing and Yap [ODY]; O'Dunlaing, Sharir and Yap [ODSY]; Reif and Sharir [RS]; Spirakis and Yap [SpY]; and Yap ([Yap84] and [Yap86]). This is by no means an exhaustive list. Much research has been done. Some books with excellent survey articles include [SHS], [SY], and [KCL].

We will not discuss this work in detail, but instead focus more on the development of the work on uncertainty.

1.4.4 Planning for Errors

The generalized spring and generalized damper approaches provided a new set of primitives with which one could reduce uncertainty in specific local settings. In parallel with this work there arose a desire to synthesize entire planning systems that could account for uncertainty. Early work considered parameterizing strategies in terms of quantities that could vary with particular problem instantiations. The skeleton strategies of Lozano-Pérez [Loz76] and Taylor [Tay] offered a means of relating error estimates to strategy specifications in detail. In particular, Lozano-Pérez's LAMA system used geometric simulation of plan steps to decide on possible motion outcomes. The simulation made explicit the possible errors that could occur. This information could be used to restrict certain parameters or to introduce extra sensing operations. Taylor's system used symbolic reasoning to restrict the values of parameters in skeleton strategies in order to ensure successful motions. Brooks [Brooks82] extended this approach using a symbolic algebra system. His system could be used both to provide error estimates for given operations, as well as to constrain task variables or add sensing operations in order to guarantee task success. Along a slightly different line, Dufay and Latombe [DL] developed a system that observed execution traces of proposed plans, then modified these using inductive learning to account for uncertainty.

1.4.5 Planning Guaranteed Strategies using Preimages

In 1983, Lozano-Pérez, Mason, and Taylor [LMT] proposed a planning framework for synthesizing fine-motion strategies. This approach is sometimes referred to as the *preimage* framework. This is because the framework generates plans by recursively backchaining from the goal. Each backchaining step generates a collection of sets, known as preimages, from which entry into the goal is guaranteed. This framework has strong connections to the dynamic programming approach mentioned above, which will be discussed further in the thesis. The preimage framework directly incorporated the effect of uncertainty into the planning process. In particular, the framework made clear how sensing operations as well as mechanical operations could be used to reduce uncertainty. An example of a mechanical operation that reduces uncertainty is a *guarded move*. During a guarded move a robot moves in the direction of an object located at an unknown distance, until contact with the object is established. Thus the uncertain location of the object becomes known with precision, relative

to the location of the robot. Guarded moves are discussed in [WG]. Earlier work using guarded moves includes [Ernst]. Mason [Mas84] showed that that the preimage planning approach is correct and bounded-complete. This means that if any system can solve a motion planning problem given the uncertainty and dynamics assumed within the preimage framework, then in fact the preimage framework will also provide a solution.

The preimage methodology spawned numerous other directions of research. Erdmann [Erd84] considered the issues of goal reachability and recognizability. He showed that for some variations of the planning problem, the task of computing preimages can be separated into two simpler problems. One of these ensures that the system will reach its goal, while the other ensures that the system will actually recognize that it has attained the goal. In general these issues are not separable. Buckley [Buc] implemented a system that computed multi-step strategies in three-dimensional cartesian space. His planner employed a discrete state graph that modelled the possible transitions and sensing operations in an AND/OR graph. Turk [Turk] implemented a two-dimensional backchaining planner.

1.4.6 Sensorless Manipulation

We have already mentioned the importance of mechanical operations for reducing uncertainty. A strong champion of such techniques is Mason. See, for instance, [Mas82a], [Mas85], and [Mas86]. In particular, Mason has looked at the problem of reducing uncertainty in the orientation of parts by pushing. Building on this work, Brost (see [Brost85] and [Brost86]) has implemented a system that can orient planar parts through a series of pushing and squeezing operations. An important aspect of these strategies is that they do not require sensing at the task level. Instead, all the actions are open loop, relying purely on the mechanics of the problem to reduce uncertainty. Other work involving the reduction of uncertainty without sensing includes the work by Mani and Wilson [MW] on orienting parts by sequences of pushing operations, the work by Peshkin [Pesh] on orienting parts by placing a series of gates along a conveyer belt, and the graph algorithms of Natarajan [Nat86] for designing parts feeders and planning tray-tilting operations. A tray-tilter is a system that orients planar parts dropped into a tray by tilting the tray. Erdmann and Mason [EM] investigated this problem, designing a planner based on the mechanics of part interactions with the walls of the tray. The planner expected as input a polygonal description of the part to be oriented along with the coefficient of friction. The output of the planner consisted of a sequence of tilting operations that was guaranteed to orient and position the part unambiguously, if such a sequence existed. A robot executed the motions suggested by the planner. This work represents a specialization of the preimage framework to the sensorless case, in which only mechanical operations may be used to reduce uncertainty. The idea for tray-tilting came from work by Grossman and Blasgen [GB] who used a combination of tray-tilting and probing operations to ascertain the orientation of a part as a prelude to grasping the part. Taylor, Mason, and Goldberg [TMG] introduced sensing back into the tray-tilter, as

a means of investigating the relative power of sensing and mechanical operations They developed a discrete planning system based on an AND/OR graph similar to the graph used in Buckley's planner.

More recent work includes the study of impact by Wang (see [Wang] and [WM]). Studying impact is of central importance, since all operations in which objects make contact involve impact. Generally, the impact occurs at scales well below those available to current sensors.

1.4.7 Complexity Results

We should mention some hardness results regarding the motion planning problem in the presence of uncertainty. Natarajan [Nat88] has shown the problem to be PSPACE-hard in three dimensions for polyhedral objects. Canny and Reif [CR] have shown the problem to be hard for non-deterministic exponential time, also in three dimensions. In general, the computability and complexity of the problem of planning in the presence of uncertainty is not known. Erdmann [Erd84] showed that the problem is uncomputable in the plane if the environment can encode arbitrary recursive functions. However, for many special cases, computable algorithms are known. Natarajan [Nat86] also has a number of results suggesting fast planning times for restricted versions of the sensorless manipulation problem. Donald [Don89] has demonstrated various polynomial-time algorithms for computing single-step strategies in the plane, assuming restrictions on the type of sensing permitted. In particular, all motions were terminated by detecting sticking in the environment. Donald also gave a single-exponential-time algorithm based on the theory of real closed fields for the multi-step strategy. Briggs [Briggs] extended these results to improve the performance of the single-step planner. Also, Canny [Can89] recently exhibited an algorithm based on the theory of real closed fields that solves the general motion planning problem under uncertainty for those cases in which the robot trajectories may be modelled as algebraic curves.

1.4.8 Further Work on Preimages

Further work on the preimage methodology has been conducted by Latombe [Lat] and his group. This work includes a study of the preimages and strategies that result from the use of various termination predicates, in addition to those used in the **LMT** preimage methodology. Others who have looked at fine motion assembly recently include [Desai], [Koutsou], [LauTh], and [Valade]. We also refer to the book [KCL] for a review of other relevant literature.

1.4.9 Guaranteed Plans

The philosophy of the preimage methodology is to generate plans that are guaranteed to accomplish some task despite uncertainty in control, sensing, and possibly the geometry of the environment. The framework assumes that uncertainty can behave

as a worst-case adversary, within specified task-dependent bounds. If a given subgoal cannot be attained with certainty assuming this worst-case behavior, then the task is deemed unsolvable. In this thesis a guaranteed strategy for solving a task will therefore refer to a set of possibly conditional actions that are certain, in the presence of this worst-case uncertainty, to accomplish the task in a bounded predetermined amount of time.

1.4.10 Error Detection and Recovery

An important offspring of the **LMT** preimage planning methodology is Donald's recent thesis (see [Don87b] and [Don89]). This work deals with the problem of representing model error and the problem of Error Detection and Recovery. The need for error detection and recovery arises naturally if one permits uncertainty in the geometric shape of objects. This is because for many interesting tasks there simply are no guaranteed plans in the sense just outlined. An example that Donald cites is the task of inserting a peg into a hole in which the size of the hole can vary due to manufacturing errors. Certainly, if the hole is smaller than the peg, then the peg cannot be inserted. Nonetheless, in many cases the hole will be large enough, and it would be foolish not to try to insert the peg. Donald claims that a robot should attempt certain tasks even if there is no guarantee of success, so long as there is a guarantee that the robot will be able to ascertain whether or not its attempt has succeeded. An error in Donald's terminology is thus more subtle than the usual notion that an error occurs when an action does not have the desired outcome. An error for which one can plan a recovery prior to execution time is not really an error, merely one of many execution-time conditions for which the system needs to check before deciding on its next action. In Donald's framework, an error is a condition of task failure for which it is impossible to plan a recovery at planning time. Thus the claim is that a robot should attempt tasks even if an error is possible, so long as the error is recognizable. Donald's formulation makes use of the preimage methodology in defining how a strategy operates. In particular, his definition of failure and the error recognizability condition are based on the preimage constructs of reachability and recognizability. These are determined by the dynamics of the task and the available sensors and termination predicates.

The important contribution of Donald's work is that it moved away from the requirement that a strategy be guaranteed to solve a task in order to be considered a strategy. This is an important and subtle point, that forms the motivation for the current thesis. By permitting strategies to fail, one can vastly increase the class of tasks that one would consider solvable. Indeed, it is clear that in some completely imperfect world, no task is ever guaranteed to be solvable assuming worst-case adversaries. The real world is such a world. Yet many tasks are solvable simply because they are attainable sometimes. Donald's thesis made this notion very precise. The aim of the current thesis is to extend some of these ideas, by considering tasks that are solvable in an expected sense. Of great importance is the ability to loop and try again, as suggested in Donald's thesis. In a worst-case sense, looping does

not help, since the strategy can always fail. However, by introducing the notions of probabilistic failure, either through actions that have probabilistic outcomes or through active randomization of run-time decisions, one can often guarantee task solvability in an expected sense.

1.4.11 Randomization

In a slightly different direction, we should mention that randomization is a technique that is sometimes used in optimizing algorithms. The *simulated annealing* approach [KGV] is a well-known technique. Roughly speaking the randomization of simulated annealing helps to avoid local minima. For any given level of randomization the system naturally converges to some subset of the state space. By reducing the level of randomization in a principled manner, this subset is made to converge to the desired optimal states. In the context of this thesis, randomization is used to avoid deterministic traps. This is similar to the avoidance of local minima. However, there is no notion of changing the level of randomization in order to ensure convergence. Indeed, for the most part we will assume that a desired goal is recognizable upon entry. More general strategies might relax this assumption, relying instead on a probabilistic prediction function to ensure that the goal is attained with high reliability.

Randomization has also been used in the domain of mobile robots. See for instance [Arkin], who injects noise into potential fields in order to avoid plateaus and ridges. [BL] have also investigated a Monte-Carlo approach for escaping from local minima in potential fields.

Some probabilistic work has aimed at facilitating the design process. For instance, [BRPM] have considered the problem of determining the natural resting distributions of parts in a vibratory bowl feeder. This information is useful for designing both part shapes and bowl feeders.

[Goldberg] is currently investigating probabilistic strategies for grasping objects. That work, in parallel with the work of this thesis, is also interested in the development of a general approach towards the analysis and synthesis of randomized strategies for manipulation tasks.

1.5 Thesis Contributions

The contributions of this thesis lie both in adding randomization to the theory of manipulation and in the practical demonstration of an assembly task using randomization. The major contributions of the thesis are:

- **Implementation of a Randomized Peg-In-Hole Task on a PUMA.** This experiment demonstrated the feasibility and usefulness of randomization in assembly operations. The sensors available to the system consisted of joint encoders on the robot and a camera positioned above the assembly. The camera was used to obtain an approximate position of the edges of the peg and the hole. These were used to suggest a nominal motion. If no edges could be obtained then

the robot would execute a randomizing motion. The system was intentionally not calibrated very well, in order to test the ability of the randomizing actions to overcome incomplete information.

- **Introduction of a Formal Approach for Synthesizing Randomized Strategies.** There exist established formalisms for generating guaranteed or optimal strategies in the presence of uncertainty. The **LMT** preimage methodology and dynamic programming are two such formalisms. This thesis builds on these approaches to include randomizing actions. Randomization is seen as another operator, called SELECT, that randomly chooses between a collection of partial strategies, under the assumption that the preconditions of at least one such partial strategy are satisfied. Partial strategies are generated by backchaining from the goal. The thesis elucidates the conditions under which this approach is expected to complete a task.

- **Analysis of a Randomized Strategy with a Biased Sensor.** The thesis presents a detailed example in which sensing error consists of a pure bias. The bias is unknown but of bounded magnitude. It is shown that a strategy which interprets the sensor as correct can fail to attain the goal. In contrast, a randomized strategy can avoid inaccurate information produced by the sensor while ensuring eventual goal attainment. Furthermore, the randomized strategy can rapidly attain the goal from certain start regions.

- **Nominal Plans.** The thesis introduces the notion of a collection of nominal plans as the choice set for a randomized strategy. This approach is a special case of the general planning methodology for synthesizing randomized strategies. The nominal plans play the role of the partial strategies in that methodology. The difference is that the nominal plans are themselves generated as guaranteed plans assuming favorable instantiations of error parameters. In particular, in this thesis, the nominal plans are strategies that are guaranteed to succeed in the absence of uncertainty. A randomized strategy tries to follow these nominal plans as well as possible despite uncertainty.

- **Progress Measures.** Nominal plans sometimes define a progress measure on state space. This is because nominal plans specify the ideal behavior of a system in solving a task. Formally, a progress measure is a real-valued function on a system's state space that is zero at the goal and positive elsewhere. Distance from the goal is a possible progress measure. If a strategy can guarantee that it makes sufficient progress on average at each point of the state space, then expected goal convergence is certain to be rapid.

- **Simple Feedback Loops.** Strategies that only consider current sensed information in making decisions are *simple feedback loops.* The thesis introduces randomized simple feedback loops. These try to make progress relative to a progress measure whenever possible and otherwise execute a random motion.

- **Random Walks.** The thesis studies random walks, as these define the most basic type of randomized strategy. A random walk forms a good model for the behavior of a randomized simple feedback loop in the presence of probabilistic errors. The thesis introduces the notion of an expected velocity as the expected change in the progress labelling of a random walk. The thesis proves that this expected velocity possesses properties similar to those of a deterministic velocity. In particular, if a strategy everywhere makes expected progress towards a goal, and if the progress measure consists of small numbers, then expected convergence to the goal must be rapid.

- **Analysis of a Randomized Simple Feedback Loop in the Presence of Unbiased Gaussian Noise.** The thesis considers a simple feedback loop for attaining a two-dimensional region in the plane. This is an abstraction of the peg-in-hole problem. The system has available to it a position sensor and a goal recognizer. The strategy is formulated assuming only that specific bounds may be placed on the error distributions that describe the sensing and control errors. Thus the strategy is known to converge eventually for all errors satisfying these bounds. For an analysis of the strategy, the sensing and control errors are each assumed to be unbiased two-dimensional normal variates. The thesis shows numerically for a particular example that the convergence properties of this randomized strategy are substantially better than those for a corresponding guaranteed strategy. In particular, the region of fast convergence is considerably greater.

- **Finite Guesses.** On discrete spaces the operator SELECT naturally only needs to guess between a finite number of possible strategies. Thus the probability of guessing an appropriate strategy is non-zero. In the continuous domain, it may be necessary to guess over an infinite number of strategies. However, the thesis shows that under suitable conditions only guesses over finite sets are required. The conditions amount to the requirement that whenever the system executes a motion for some time, the predicted possible locations of the system at that time form an open set.

- **Near-Sensorless Tasks** are defined as tasks in which there is no sensing except to signal goal attainment. Blindly inserting a key into a hole is one such task. The thesis shows that in this context there are tasks for which there exist guaranteed solutions that require an exponential amount of time to execute in the worst case, while there exist randomized solutions that require a polynomial amount of time to attain the goal, in an expected sense. This result demonstrates that randomization need not necessarily increase convergence times.

1.6 Thesis Outline

Chapter 2 presents a more detailed outline of the thesis. This chapter also contains further motivational material. The chapter is intended both as a second introductory chapter and as a precis of the thesis.

Chapter 3 develops the basic approach. This is done in the discrete setting, for simplicity. Fortunately, many of the results carry over to the continuous domain. The basic idea is to use the traditional methodology for computing guaranteed plans as a means of suggesting partial or nominal plans. Sensing uncertainty may prevent the system from satisfying the preconditions of any particular nominal plan. However, in some cases the system can readily satisfy the union of all the plans' preconditions. Then it makes sense for the system to randomly and repeatedly choose and execute a nominal plan. The hope is that the system will eventually choose a plan whose preconditions are satisfied, and which therefore will successfully accomplish the task. Chapter 3 considers the conditions under which this type of strategy may be applied. Of particular interest are tasks in which there is a progress measure. If the system can locally make progress on the average, then the overall expected convergence time may be bounded readily.

Chapter 4 extends this approach to the continuous domain. Some subtleties enter into the picture. In particular, in order to be certain of eventual convergence, a randomized strategy should only make guesses that have a non-zero probability of success. Given an infinite number of nominal plans, as is possible in the continuous domain, the probability of guessing correctly may actually be zero. Chapter 4 examines this problem and shows that often it is reasonable to consider only a finite number of nominal plans.

Chapter 5 analyzes the task of moving a point on the plane into a circle, in the presence of sensing and control uncertainty. This is a natural generalization of the peg-in-hole problem considered at the beginning of the thesis. The analysis involves an approximation by a diffusion process that establishes fast convergence times for a range of goal sizes.

Chapter 2

Thesis Overview and Technical Tools

The purpose of this chapter is to provide a basic overview of the thesis. The chapter is intended both as a self-contained summary of the thesis as well as a guideline for the results presented in the remaining chapters. We will motivate the basic problem, present the technical tools and definitions, and mention the main results of the thesis. All this will be done at a fairly high level, with the details of the definitions and proofs left for future chapters. It is hoped that the early presentation of the main issues will provide a cohesive guideline for the more technical points of the later chapters.

The first major section of this chapter provides a high-level perspective and motivation. The second section is concerned with basic definitions. Towards the end of the section we introduce the notion of randomized strategies. The third major section of the chapter presents a detailed example that is intended to highlight the importance of randomized strategies. Finally, the last section discusses in some more detail the particular focus on randomized strategies taken by this thesis.

2.1 Motivation

In general one should think of randomization as a primitive strategy, and thus as a tool at the lowest level. One should not forget all the work on the synthesis of strategies for solving tasks in the presence of uncertainty. Instead, randomization should be viewed as an operation that is superimposed on top of the work for generating guaranteed strategies. Indeed, randomization is even physically superimposed on top of these strategies. **It is the combination of sensing, mechanics, and randomization that achieves a task, not any one of these alone.** We will study, primarily in chapter 3, strategies that judiciously make use of sensing, predictive ability, and randomization. The physically realizable solutions to tasks are those for which on the average progress is being made towards the goal. The randomization ensures that partially modelled system parameters may be ignored, while the sensing and task mechanics ensure that progress is made towards the goal whenever the randomization

has placed the system in a fortuitous position.

2.1.1 Domains of Applicability

The broadly intended domains of applicability for the material presented in this thesis are:

- Parts Assembly and Manipulation.
  - In the presence of sensing and control uncertainty.
  - In environments with sparse or incomplete models.
  - During the fine-motion phase of tight assemblies.
  - For parts orientation and localization.

  (And combinations of these scenarios.)
- Mobile Robot Navigation.
  - With noisy sensors.
  - In uncertain environments.
- Facilitate Design.
  - Of special purpose sensors useful for solving particular tasks.
  - Of parts shaped to permit easy mechanical assembly.

The main focus of the thesis is within the first domain on this list. This domain consists of tasks involving the assembly and manipulation of parts. Examples include the mating of two or more parts, the grasping of a part, and the orienting and localization of one or more parts whose initial configurations are unknown. By localization we mean the constraining of a part's configuration in a purposeful manner, possibly as a prelude to some other operations. The archetypical example of a parts mating operation is given by the task of inserting a peg into a hole. This is a classic example, yet its generality remains. This generality stems from the observation that almost any assembly involving rigid or nearly-rigid bodies may be viewed locally as a peg-in-hole assembly. The tasks of grasping and orienting parts are themselves fundamental to manipulation. In order to assemble two parts, these must be located and manipulated. The manipulation may involve grasping or it may involve impact operations, such as pushing or hitting. In some broad sense grasping subsumes these latter operations, as they occur naturally at some scale during any operation involving the contact of two or more objects. Finally, parts ultimately must be oriented and localized in order to be assembled. A system need not necessarily be cognizant of the localization operation, yet localization must occur at either the mechanical or sensing

levels. More generally, a task that involves the transfer of objects from a state of high entropy to an assembled state, such as the task of picking a part out of a bin containing several different randomly oriented parts, determining the part's pose, and then placing it in some constrained locale, requires variations of all of these operations. In particular, almost by definition, such a task requires considerable localization.

Most of the results of the thesis will be developed with the inspiration of these examples in mind. However, the results are sufficiently general that they may be applied to domains other than pure manipulation. Some of these are indicated in the list above.

2.1.2 Purpose of Randomization

One of the key motivations for considering randomized strategies is given by our description of manipulation tasks in the presence of uncertainty as methods for reducing entropy. Specifically, parts are moved from a disorganized state into an assembled state, from an unknown orientation to a known orientation, from an unconstrained location to a grasped location, and so forth. Reducing entropy is generally difficult, requiring considerable information about the world. Randomization permits the view of an organized state as simply one of many random states. By actively randomizing, a system can under suitable conditions ensure that it will eventually pass through this desired state. (The suitable conditions effectively postulate lower bounds on the probability of success.)

Standard approaches for solving tasks that involve the reduction of uncertainty include:

- Perfection.
  - Model the world perfectly.
  - Reduce sensing errors to zero.
  - Reduce control errors to zero.
- Plan for Uncertainty.
  - Use sensing when possible to gain information from the environment.
    - For instance, use a combination of position and force sensors in order to gain more information than either sensor could provide in isolation. As an example, a force sensor might register contact with a table, while a position sensor could localize that contact to within some small range.
    - Build special sensors to detect particular system states. This includes light beams at finger tips, touch sensors, special calibration devices, lasers, structured light, and so forth.
  - Use the mechanics of the domain to reduce uncertainty.
    - For instance, bump into an object in order to reduce the uncertainty of the relative position of that object.

    * Drop a polyhedral part onto a table in order to reduce its orientations to a manageable number.
    * Design parts and feeding/assembly devices concurrently, with the aim of simplifying the grasping or localization process.
  - Strategically combine sensing and action.
    * For instance, in order to move one part within a certain distance of another object whose location is unknown, it makes sense to first bump the part into that object, then back away by the desired distance, if possible.

- Tolerate Failure.
  - Give up the insistence on a guaranteed strategy as the only means of solving a task.

Accepting Uncertainty

The assumption that the world is perfect is much too strong an assumption to be realistic. Instead, as we outlined in section 1.4, much effort has been devoted over the last few decades to accounting for uncertainty explicitly. The aim has been to reduce uncertainty or entropy by judicious use of sensing and action. The difficulty with such approaches is that they tend to make strong assumptions about the world. For instance, generally those frameworks that produce guaranteed plans have trouble dealing with tiny variations in geometry. A strategy that slides one object on top of another may fail if the component surfaces contain small nicks and protrusions. Similarly, if a sensing error is larger than expected, or if a sensor contains an unknown bias, a strategy that relies crucially on the validity of its assumptions will fail. This defeats the philosophy motivating the construction of planners that explicitly account for uncertainty. That philosophy states that one should from the outset be aware of uncertainty, rather than ignore it in the hope that the plans developed for a perfect world will be good enough in the face of uncertainty. The philosophy is defeated because the strategies developed in the quest of guaranteed plans are only as good as the assumptions preceding them. Of course, everyone is aware of this dependence, yet it lingers. More importantly, the dependence can lead to the desire to model the world accurately, to improve one's sensors, and to improve one's control systems, solely for the sake of solving a particular task more easily. These are highly worthwhile goals, but they run the risk of ignoring a set of crucial intellectual questions:

- What is the information needed to solve a task?
- What tasks can be solved by a given repertoire of operations?
- How sensitive are solutions of tasks to particular assumptions about the world?

Indeed, the design of better systems for dealing with uncertainty should be interwoven with the investigation of these questions. The answers to these questions will themselves facilitate the design of better systems for dealing with uncertainty and will improve planning technologies.

A key approach listed above is that of tolerating failure. This is a fairly recent idea within the formal planning methods of robotics (see [Don87b]). It is important because it reminds us of the right psychological framework. No task possesses an absolutely guaranteed solution. Instead of searching for guaranteed solutions, one should try to answer the three questions above, for any task of interest. There is a spectrum of assumptions, a spectrum of strategies, and a corresponding spectrum of outcomes for any given assumptions and strategy. Failure is always one of the possible outcomes in this spectrum. The question is, under what assumptions?

Clearly, the work on uncertainty over the past several decades has been trying to answer the three questions. They remain unanswered in generality. This thesis is one further attempt to look at a particular aspect of the answer to these questions.

Randomization is Everywhere

Randomization enters into the investigation of these questions at the simplest level. In some sense randomization is omnipresent. For instance, uncertainty that is due to noise, either in sensing or control, may be thought of as randomization on the part of nature. The basic issue that this thesis begins to address is how active randomization on a robot's part can aid in the solution of tasks.

Some advantages of randomization are:

- Increase the class of solvable tasks.
- Reduce the dependence of task solutions on assumptions about the world.
- Simplify the planning process.

We will discuss these properties more throughout the thesis. In brief, the class of solvable tasks is increased because the class of strategies is enlarged beyond the class of guaranteed strategies. Recall that a guaranteed strategy is certain to accomplish a task in a bounded predetermined number of steps. Randomization increases the class of solvable tasks because the class of randomized strategies includes strategies whose success is not guaranteed on any particular step, but merely in an expected sense. Randomization decreases dependence on assumptions when it ensures that a system will eventually behave in a manner compatible with unknown or unmodelled parameters. There are limits, of course, such as trap states or degenerate goals that must be avoided by any strategy. Finally, planning is simplified whenever a planner may substitute a simple randomized strategy in place of a possibly complicated guaranteed strategy. For instance, a random walk is a simpler strategy than a spiral search. It requires less history, although it may require more time to converge to a desired goal region.

Eventual Convergence

In a sense we may think of randomization as a means of traversing the state space in a blind manner. Thus randomization forms the most primitive of strategies for solving a task. By performing a random walk in state space, the system will, under suitable conditions, eventually pass through the goal. The suitable conditions amount to guaranteeing a minimum probability of success.

Of course, there are some disadvantages to randomization. If manipulation tasks may indeed be thought of as means of reducing entropy, then randomization seems inappropriate. Indeed, one would expect randomization to increase entropy. However, this is not always the case. Furthermore, it says merely that one might have to wait a long time before the system attains a goal. Other difficulties arise in ensuring that the randomization actually covers the space of interest, that is, that the goal is reachable. A third difficulty arises in terminating a strategy. Somehow there must be appropriate information that enables a system to recognize or predict goal attainment. All these issues will be dealt with in the thesis.

Fast Convergence

Of particular interest is the question of convergence times. It clearly would be inappropriate to try to insert a peg with six degrees of freedom into a hole using purely random motions. The hole forms a relatively small region within the six-dimensional configuration space of the peg. Finding this region without any sensing from far away would require an unreasonable amount of time. However, if one can bring the peg close to the hole using available sensors, then one can reduce the space that must be searched. If one can also remove some of the peg's degrees of freedom by making contact with portions of the hole, then one can further reduce the space that needs to be searched, by reducing its dimensionality.

2.2 Basic Definitions

This section defines the basic tools used by the thesis. This includes the spaces of interest, the representation of uncertainty, and the types of strategies explored throughout the thesis.

2.2.1 Tasks and State Spaces

A task is modelled as a problem on some state space. The state space may be discrete or continuous. The state space should consist of all the parameters of a system that are required to predict its future behavior. In other words, knowing the current state of the system and some action applied to the system, it should be possible to predict the resulting state or states of the system without reference to past states.

A task is specified as the attainment of some goal region in state space. Sometimes a starting region may be specified as well.

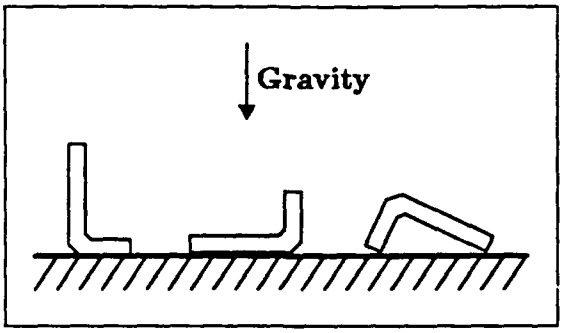


Figure 2.1: This figure indicates three stable configurations of a planar Allen wrench lying on a horizontal table. These configurations may be used to define a discrete state space.

We should mention briefly that the *configuration space* [Loz83] of a system is the space describing the degrees of freedom of the system. For instance, the configuration space of a rigid object in three dimensions is a six-dimensional space corresponding to three translational and three rotational degrees of freedom.

The relationship between the state space and the configuration space of a system depends on the dynamics of the system. For simplicity, we often assume that the dynamics are first-order and that the future state of the system can be predicted from its current configuration and an applied velocity. In that sense the state space and the configuration space are identical. We will thus often not distinguish between the two representations, although it should be understood that this is not sufficient if the dynamics are of a higher order.

Continuous Space

An example of a task specified in a continuous space is given by the peg-in-hole problem of section 1.1. The relevant state space for that problem is a three-degree-of-freedom space, consisting of two translational and one rotational degrees of freedom. Actions are specified as changes in position and orientation. The goal is the range of positions and orientations for which the peg is directly over the hole. This is a fairly small volume in the three-dimensional state space.

In general in this thesis we will assume that a continuous state space is some bounded subset of $\Re^n$, for appropriate dimension n. Such a space corresponds naturally to a system with several translational degrees of freedom, but no rotational degrees of freedom. However, natural generalizations to n-dimensional manifolds exist. See, among others, [Loz81], [ScShII], and [Can89].

Discrete Space

An example of a discrete state space is given by the stable orientations of a polyhedral part resting on a horizontal table under the influence of gravity. Figure

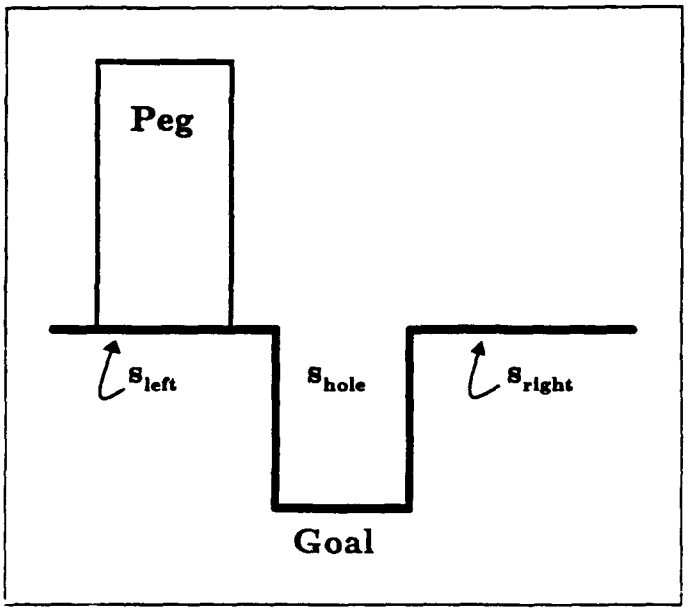


Figure 2.2: A two-dimensional peg-in-hole problem. Also shown are three states that might be used in a discrete approximation to the continuous problem.

2.1 depicts the planar case. The figure shows three stable orientations of a planar part resting on a horizontal table. By tilting the table for a short amount of time the part can be made to roll between different such configurations. While the analysis of the forces required to move the part may require consideration of a continuous space, once this analysis has been performed, it is sufficient to consider the resulting discrete space in planning operations to orient the part stably. This example is taken from [EM].

Discrete representations also arise as approximations to continuous spaces. For instance, one might place a fine tiling over a continuous state space, then regard each of the tiles as a state in a discrete state space. Finally, sometimes tasks formulated in continuous spaces may be transformed naturally into a discrete representation. For instance, consider the planar task of inserting a two-dimensional peg into a two-dimensional hole (see figure 2.2). Assume that the peg can only translate, but not rotate. If the peg has made contact with the horizontal edges near the hole, then the problem can be represented as a three-state system. One state corresponds to contact with the edge to the left of the hole, another state corresponds to contact with the edge to the right of the hole, and the third state corresponds to entry into the hole. While this representation discards some information, such as the distance of the peg from the hole, it still retains the basic geometrical relationships required to attain the hole.

The discrete spaces treated in this thesis are assumed to be finite. Thus a discrete

state space is simply a finite set $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_n\}$, for some n. Most of the development of the theory of probabilistic strategies will be done on finite discrete spaces (see chapter 3). This is primarily a device for simplifying the presentation. The results carry over with appropriate modifications to continuous spaces. The extension to continuous spaces is handled in chapter 4.

2.2.2 Actions

Actions are transformations on the state space. There are three broad classes of actions: *deterministic*, *non-deterministic*, and *probabilistic*. In some sense, the category of non-deterministic actions includes deterministic and probabilistic actions as special cases. Another special case is given by non-deterministic actions whose underlying non-determinism is constrained. These actions fall under the category of *partial adversaries*, which we discuss below as well.

In terms of information content, the ordering of action categories by decreasing certainty is: DETERMINISTIC, PROBABILISTIC, PARTIALLY ADVERSARIAL, NON-DETERMINISTIC.

Deterministic Actions

A deterministic action maps each state of the state space to some other state. This is most easily represented in the discrete case. If $s \in \mathcal{S}$ is a state, and A is an action, then $A(s)$ is some other state in $\mathcal{S}$. For instance, in the three-state peg-in-hole example of figure 2.2, an action might correspond to the operation MOVE-RIGHT. Denote the three states by s_{right}, s_{left}, s_{hole}, corresponding to contact with the edge to the right of the hole, contact with the edge to left of the hole, and entry into the hole, respectively. Then one might have that MOVE-RIGHT$(s_{\text{right}}) = s_{\text{right}}$, MOVE-RIGHT$(s_{\text{left}}) = s_{\text{hole}}$, and MOVE-RIGHT$(s_{\text{hole}}) = s_{\text{hole}}$.

In the continuous case, executing an action generally entails performing some operation over some duration of time. For instance, for a simple first-order linear system, an action may correspond to executing a velocity over some time interval. In that case, if $\mathbf{x} \in \Re^n$ is a state of the system, then an action is of the form $(\mathbf{v}, \Delta t)$, and the effect of an action is to move $\mathbf{x}$ to the state $\mathbf{x} + \Delta t\, \mathbf{v}$.

Non-Deterministic Actions

A non-deterministic action is a relation on the state space rather than a function. It transforms each state to a set of states. The purpose of a non-deterministic action is to model uncertainty. This may correspond either to non-determinism in the transitions specified by the action, or it may simply correspond to a paucity of knowledge in modelling these transitions. In the discrete case we will write the effect of a non-deterministic action as $F_A(s)$. This is called the *forward projection* of the state s under the action A. The forward projection is a subset of the state space. A similar representation exists for the continuous case, although now the action must also include a time parameter.

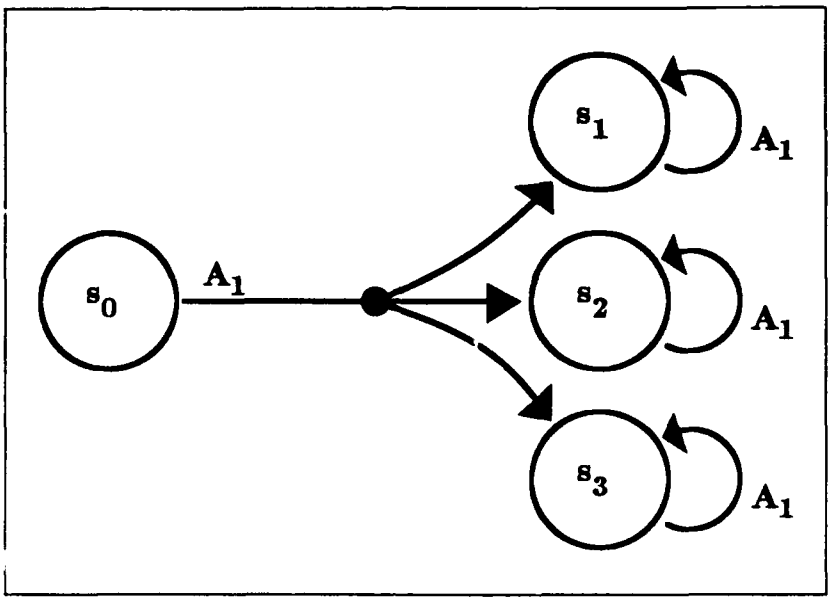


Figure 2.3: Graphical representation of a non-deterministic action A_1.

Figure 2.3 depicts a four-state system, in which action A_1 non-deterministically maps state s_0 to the three other states. In other words, $F_{A_1}(s_0) = \{s_1, s_2, s_3\}$.

A non-deterministic action measures the worst-case behavior of the system. Nothing is said about the actual likelihood that a particular transition will be taken. In other words, if a state s_f appears in the set $F_A(s)$, then one must assume that action A might cause state s to move to state s_f. However, one cannot be sure that this will ever occur.

One view is to imagine an adversary, who can force state s to move to state s_f whenever this would be to one's disadvantage, but who also can move s to some other state in $F_A(s)$ whenever one would actually like to attain s_f. This is what is meant by a worst-case modelling of an action.

Partial Adversaries

As we have indicated, the non-deterministic representation of actions provides a worst-case view which may considerably overestimate the uncertainty in the actions. For instance, consider a first-order linear system in $\Re^2$, governed locally by the equation $\mathbf{x}(t) = \mathbf{x}_0 + t\,\mathbf{v}$, where $\mathbf{x}_0$ is the starting state, $\mathbf{v}$ is the actual velocity of the system, and t is the elapsed time. Suppose in fact that the starting state is the origin, and that the action consists of commanding the nominal velocity $(1, 0)$ for some time interval Δt. Suppose that the effect of this action is modelled non-deterministically. In particular, any velocity of the form $(1, \epsilon)$ can result, where $\epsilon \in [-0.25, 0.25]$. Now imagine that one repeatedly commands this action, say 1000 times, each time for duration $\Delta t = 1$. The non-deterministic representation says little about the actual

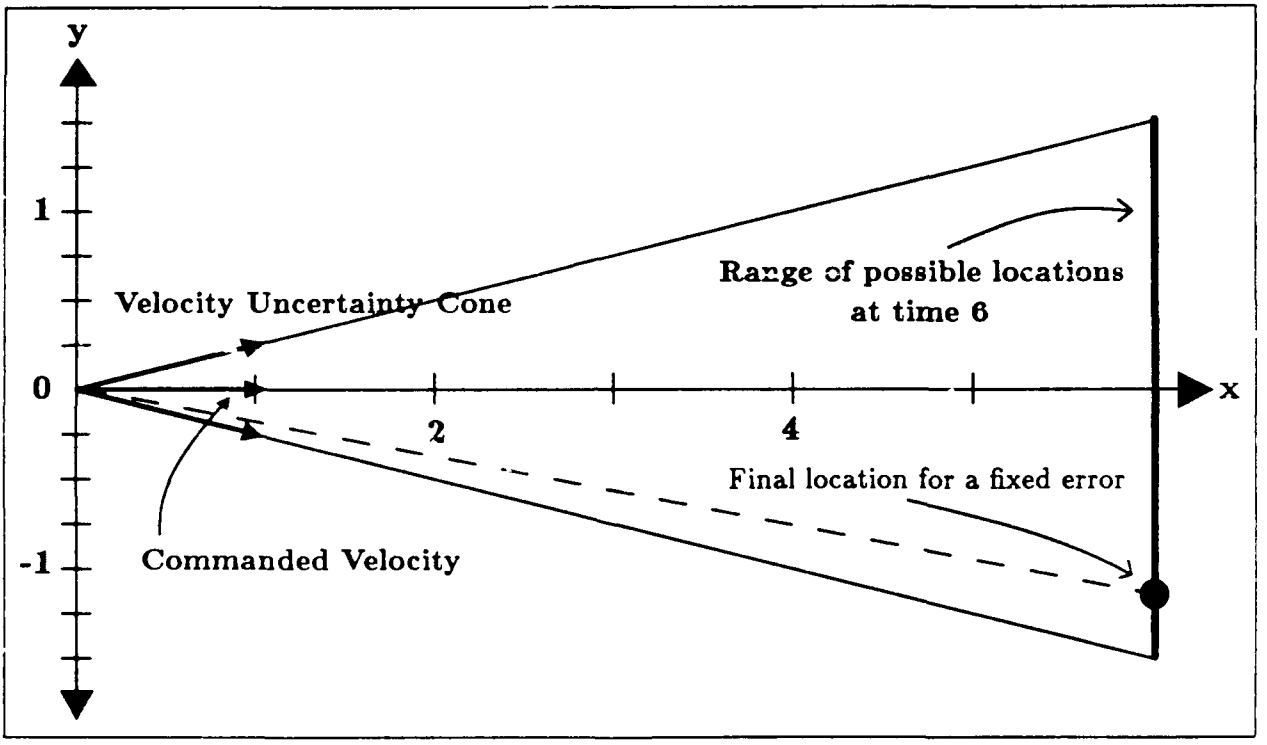


Figure 2.4: This figure shows the possible locations of the system after executing a commanded velocity subject to uncertainty for 6 time units. The commanded velocity is $(1, 0)$. The effective velocity is given non-deterministically by $(1, \epsilon)$, with $\epsilon \in [-0.25, 0.25]$. The figure also shows the final location if the error ϵ is fixed. In this case the resulting motion is repeatable.

location of the system after these 1000 actions. All one can say for sure is that the x-position will be 1000, while the y-position will lie in the range $[-250, 250]$. See figure 2.4 for the state of the system at $t = 6$.

Indeed, if an adversary could at each instant in time choose ϵ arbitrarily within the range $[-0.25, 0.25]$, then this is the best possible prediction of the future state of the system. Yet, it may turn out that the system cannot actually behave in this worst-case manner. In particular, the non-deterministic representation of the velocity as $(1, \epsilon)$ may be due to a fixed but unknown bias in the control system. Thus, after executing the velocity for time $t = 1000$, the system is actually at the location $(1000, 1000\epsilon)$, with **fixed** $\epsilon \in [-0.25, 0.25]$. Offhand, this case may not seem any better than before; the prediction of the final state of the system again places the y-coordinate somewhere into the range $[-250, 250]$. However, if one could make observations of the system's position at some time after initiating the motion, then one could accurately predict the final location of the system. More importantly, the action is repeatable. In other words, whenever the system starts at the origin, subject to the commanded velocity $(1, 0)$ for time $t = 1000$, the system will wind up at the location $(1000, 1000\,\epsilon)$, where ϵ is some **fixed** number in the range $[-0.25, 0.25]$.

One way to view the previous example is to realize that the non-deterministic choices possible at any instant in time are coupled. In this example, nature cannot choose the velocities arbitrarily at every instant in time. Instead, the fixed bias constrains these choices over time. Only the bias itself is arbitrary and unknown. Said differently, the underlying uncertainty does not behave like a worst-case adversary, but merely like a partial adversary. Choices made by the adversary constrain further choices. From a predictive point of view one may still wish to model the system in a worst-case manner. However, one can often take advantage of the coupling between the unknown parameters of the system, without initially knowing the instantiation of these parameters. In the previous example this advantage takes the form of being able to execute an action repeatably. We will demonstrate another example involving sensing biases in section 2.4.

One should realize that this is a particularly simple example. In general there may be several components to an error. Some of these may behave adversarially, some may behave like partial adversaries, and some may behave probabilistically (see the next paragraph). For instance, it is quite common to have an error that consists of biased noise. In this case the bias is like a partial adversary, while the noise is probabilistic.

Probabilistic Actions

Probabilistic actions are a special case of non-deterministic actions, in which it is possible to assign a probability density function to the forward projection. Consider in the discrete case the forward projection $F_A(s)$ of some state. This set is of the form $F_A(s) = \{s_1, \cdots, s_q\}$, for some set of states $s_1, \ldots, s_q$. For a probabilistic action A, one can assign to each state s_i a probability p_i. This means that if the system is initially in state s, and one executes action A, then state s_i will be attained with

probability p_i.

A probabilistic representation of an action carries with it considerably more information than does a non-deterministic model. Clearly not all actions may thus be modelled. For instance, in the example of figure 2.4, if the error in the commanded velocity is indeed a fixed but unknown bias, then one cannot model it as a probabilistic action. However, if the error is due to noise, with a known bias, then it makes sense to think of the error ϵ as a random variable in the range $[-0.25, 0.25]$. In that case, the extra information provided by the probabilistic representation manifest itself via the central limit theorem. In particular, suppose that the basic action consists of commanding the velocity $(1, 0)$ for time $\Delta t = 1$. Now imagine applying this action 1000 times consecutively. Then the central limit theorem tells us that the y-coordinate of the final position of the system will be normally distributed about $1000\mu_\epsilon$. Here μ_ϵ is the expected value of ϵ, that is, the bias in the noise.

2.2.3 Sensing

Sensing aids in reducing uncertainty. A system that observes its behavior can sometimes compensate for errors in control. However, uncertainty enters into sensing as well. We will consider a spectrum of sensing uncertainty, analogous to the various forms of action uncertainty. Specifically, of interest are *perfect sensing*, sensing with *probabilistic errors*, sensing with *non-deterministic errors*, and *sensorless* systems, that is systems with infinite sensing uncertainty. Closely related to the sensorless systems are *near-sensorless* systems, in which there is just enough sensing to detect task completion.

In terms of information content, the ordering of sensing categories by decreasing certainty is: PERFECT, PROBABILISTIC, NON-DETERMINISTIC, NEARLY-SENSORLESS, SENSORLESS. As with control uncertainty, there are also PARTIALLY ADVERSARIAL versions of non-deterministic sensing.

Perfect Sensing

A perfect sensor is one that reports the system's state with complete accuracy. It is fairly easy to plan strategies for such systems, even if control is uncertain. We shall discuss this issue further below.

Imperfect Sensing: Basic Terms

An imperfect sensor is a sensor that returns a sensed value that need not be the actual state of the system. Generally, given a state x of the system, there is a collection of sensor values $\{x^*\}$ that might be observed. For each sensed value x^*, the system can infer that the actual state of the system must lie in some set of interpretations $I(x^*)$. The exact nature of the interpretation set depends on the type of sensor.

The next few paragraphs discuss imperfect sensors in more detail, as well as provide examples of such sensors.

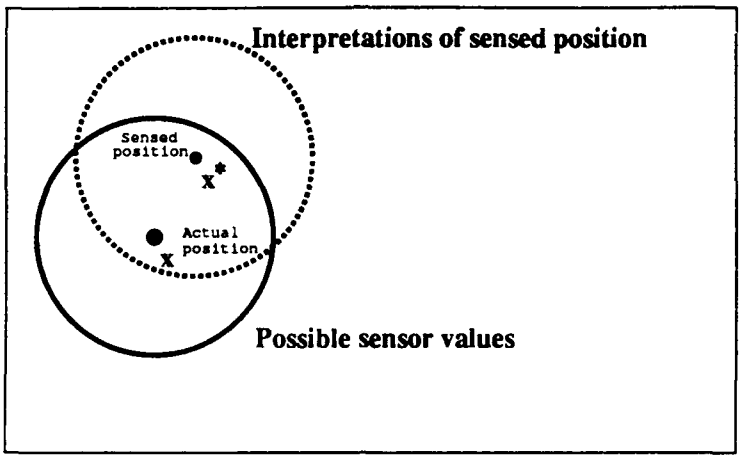


Figure 2.5: This figure shows the actual location x of a system, along with an observed sensor value x^*. The disk bounded by the solid circle depicts the range of possible sensor values assuming a bounded but unknown sensing error. The disk bounded by the dashed circle depicts the possible interpretations of the observed sensor value. Notice that the actual state of the system is indeed a possible interpretation of the observed sensor value.

Imperfect Sensing: Non-Deterministic Sensing

In the non-deterministic case, for each actual state x of the system, there is a collection $\Xi(x) = \{I(x^*)\}$ of possible interpretation sets that might result upon sensing. There is one interpretation set $I(x^*)$ for each possible sensor value x^*. No further assumption is made about the actual likelihood of observing a particular sensor value x^*. This is analogous to the worst-case representation of uncertainty in actions. Similarly, each interpretation set $I(x^*)$ is a set of possible states of the system. Again no assumption is made about the actual likelihood that the system is in a particular state in the set $I(x^*)$, given that x^* has just been observed.

As an example, imagine that the state space is a subset of the real line. Suppose that whenever the actual state of the system is at the point x, then the range of sensor values that the system might observe is given by the interval $(x - \epsilon, x + \epsilon)$ for some $\epsilon > 0$. This is sometimes referred to as an *unknown but bounded* model of uncertainty. Clearly, if the system observes a sensor value x^*, then the set of interpretations of that sensor value is given by the interval $I(x^*) = (x^* - \epsilon, x^* + \epsilon)$. Figure 2.5 depicts a two-dimensional example.

Imperfect Sensing: Probabilistic Sensing

A probabilistic sensor is an imperfect sensor for which there exists a probability density function over the range of possible sensor values. For instance, given a

position $x \in \Re^n$, the range of sensor values x^* might be described by a normal distribution centered at x. Inverting this collection of distributions using Bayes' rule allows one to construct for each sensor value x^* a set of interpretations $I(x^*)$. This set of interpretations is itself a probability density function describing the likelihood that the system is in state x given that one has observed sensor value x^*.

Imperfect Sensing: Sensorless and Near-Sensorless Tasks

In sensorless tasks there is no sensing, whereas in near-sensorless tasks there is no sensing except to signal goal attainment. Without sensing a system must rely entirely on its actions and predictive ability to attain the goal. In the near-sensorless case this is essentially true as well, except that there is an additional bit of information which signals success should the goal ever be attained. This is useful for systems that repeatedly execute a loop that has some chance of attaining the goal but that is not guaranteed to attain the goal. See below. We prove later (see section 3.13.2) that the class of tasks solvable using a sensorless system is very much like the class of tasks solvable using a near-sensorless system. Of course, for any particular task, adding a goal recognizer can change the task from being unsolvable to being solvable.

Any open-loop task is by definition a sensorless task. For instance, the gross motions used to manipulate objects in uncluttered environments are examples of sensorless tasks. Within the fine-motion phase of assembly an example of a sensorless task is the process of orienting parts by pushing one part against another. This is similar to the palletizing that occurs when for instance luggage containers are loaded onto airplanes. The containers are rolled onto large loading lifts that lift the containers from ground level up to the cargo door of a plane. The containers are generally not yet oriented properly after having been rolled onto the loading lifts. However, the platform of the loading lift consists of motorized wheels that push the container into a corner of the lift assembly. The result is that the the container is oriented properly in the absence of any sensing. Many feeder mechanisms operate on this principle. [Mas85] refers to such operations as *funnels*. Indeed, a funnel for filling a jar with water or flour is a classic example of a strategy that uses task mechanics rather than sensing to constrain the behavior of a system.

More generally, many operations involve aspects of sensorless strategies. This is because often some mechanical interaction between parts occurs below the resolution of available sensors. The motion of an object due to impact during a gasping operation is one example.

Examples of near-sensorless system can easily be constructed from examples of sensorless systems. Essentially the goal recognizer acts as a verification mechanism that ensures that the task really has been accomplished. This is useful particularly when one's assumptions about the task mechanics are subject to uncertainty.

In the context of this thesis, an example of a near-sensorless system is given by the behavior of a randomized strategy such as the peg-in-hole strategy of chapter 1, once the sensors no longer provide useful information to guide the assembly. Essentially the strategy is operating without any relevant sensing. However, the goal recognizer is

used to terminate the strategy. In the peg-in-hole case, goal recognition was achieved by noting that the camera image indicated that the peg had entered the hole.

2.3 Strategies

Of great importance is the process by which one synthesizes strategies to the various types of tasks discussed above. Part of the question is the definition of a strategy.

2.3.1 Guaranteed Strategies

Traditionally, *guaranteed strategies* and *optimal strategies* have been the focus of attention. These in turn may be subdivided by the manner in which they treat sensory and predictive information. At one extreme is a strategy that makes full use of *sensing history* and *forward projections* of the current state. At the other extreme is a *simple feedback loop*, which is a strategy that only considers current sensory information in making decisions.

Recall that by a guaranteed strategy we mean a set of possibly conditional actions that are certain to accomplish a task in a bounded predetermined amount of time.

2.3.2 Randomized Strategies

This thesis introduces a class of strategies complementary to guaranteed strategies, known as *randomized strategies*. One of the characteristics of a guaranteed strategy is that it attains its goal in a bounded predetermined number of steps. In contrast, a randomized strategy consists of a sequence of operations that only has some non-zero probability of attaining its goal. The key to success with a randomized strategy is to place a loop around this sequence of operations. This means that one repeatedly executes the sequence of operations inside the loop until the sequence eventually succeeds.

A key ingredient to randomized strategies is active guessing or randomization. This takes the form of either guessing the location of the system or of executing an action that has been randomly selected from some applicable set of actions. Guessing the location of the system is a means of compensating for uncertain sensing information. Executing a random action is a means of avoiding getting stuck in some location from which there is no guaranteed strategy of escape. Clearly one may draw connections between these two forms of randomization.

The motivation for considering randomized strategies is to increase the class of solvable tasks, to reduce knowledge requirements, and to simplify the planning process. This is facilitated in two ways. First, by not insisting on guaranteed plans, one automatically broadens the class of tasks for which one can provide solutions, although the solutions are now solutions in a probabilistic sense. Second, by actively randomizing at both the sensing and action levels, one can reduce the knowledge details needed to solve a task. This makes it easier to plan solutions to tasks for

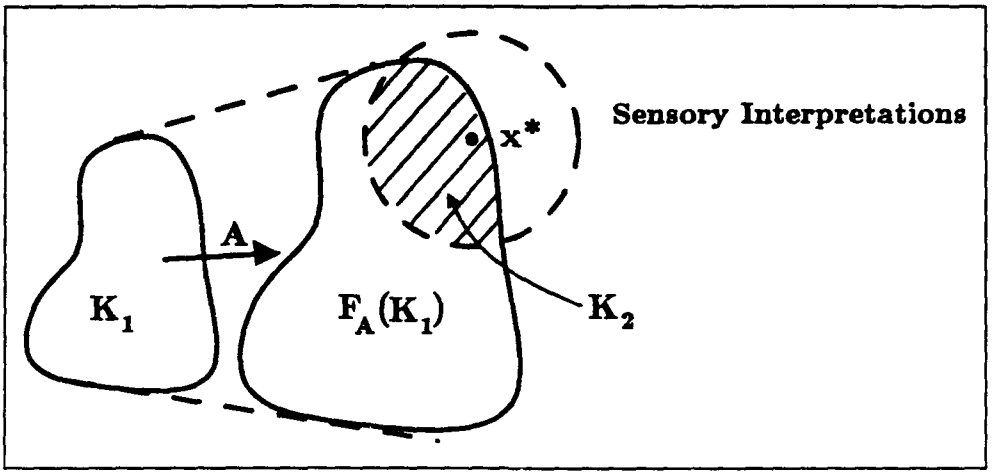


Figure 2.6: This figure depicts schematically how a system might update its run-time knowledge state using both prediction and sensing. First, the system forward projects the previous knowledge state K_1 using the current action A. Second, the system intersects the resulting set $F_A(K_1)$ with the interpretations of the current sensed value x^*. K_2 is the updated knowledge state.

which there exist guaranteed solutions. In addition, it permits some tasks, for which there are no guaranteed solutions, to be solved in an expected sense. In effect, randomization blurs the details of the environment. For instance, in the peg-in-hole problem of figure 2.2, if the horizontal edges contain slight nicks, then the peg could become stuck while sliding. Rather than plan for every possible nick explicitly, it makes sense to invoke some type of randomizing action that is likely to start the peg sliding again.

2.3.3 History and Knowledge States

We mentioned above that strategies may be classified by their use of history. This applies both to guaranteed strategies and to randomized strategies. Another way to phrase this is to characterize the knowledge state of the system at run-time. A knowledge state is always some subset of the state space. It reflects the certainty with which the system knows its actual state. In the case of perfect sensing, the knowledge state is a singleton set containing the actual state of the system. More generally, a knowledge state can be an arbitrary subset of the state space.

Systems differ in the manner by which they update their knowledge states. A *simple feedback loop* only considers current sensed values. Thus the knowledge state of a simple feedback loop is always the most recent sensory interpretation set $I(x^*)$, where x^* is the most recently observed sensor value.

A system that makes full use of sensing history updates its knowledge state by

forward projecting the previous knowledge state and intersecting it with the current sensory interpretation set. We will state this semi-formally for the discrete case, in the next paragraph. A similar description exists for the continuous case; it is depicted pictorially in figure 2.6. Both these descriptions apply to non-deterministic actions and non-deterministic sensing. In the probabilistic setting, the analogous operation is given by the Kalman filter (see [Brown], for instance).

Turning now to the discrete case, suppose that the most recent knowledge state is K_1, that the action just executed is A, and that the sensory interpretation set is I. The new knowledge state derived from this information is given by $K_2 = F_A(K_1) \bigcap I$. In other words, the previous knowledge state is first forward projected to account for any changes due to the executed action. The resulting set is then intersected with the sensory information. Updating the knowledge state in this manner on each time step ensures that full use is made of sensing history and of predictive ability, within the bounds given by the non-deterministic description of sensing and action uncertainty.

2.3.4 Planning

Planning Guaranteed Strategies

Once one has the notion of a knowledge state, planning guaranteed strategies is conceptually simple. Specifically, one backchains in the space of knowledge states, starting from the goal. This process is sometimes referred to as *dynamic programming.* It is discussed in further detail for the discrete context in section 3.2.4. Chapter 4 discusses the [LMT] preimage framework, which is a backchaining approach for computing guaranteed strategies.

Briefly, backchaining proceeds as follows. Given a collection of goal states $\{G_\alpha\}$, the planner determines all pairs of knowledge states and actions (K, A), for which attainment of one of the goals G_α is guaranteed. This means that for each sensory interpretation set $I(x^*)$ that the run-time system might observe upon execution of action A, the updated knowledge state lies inside a goal. Formally one must have that $F_A(K) \bigcap I(x^*) \subseteq G_\alpha$ for some α. The collection of all knowledge states K that satisfy this condition comprises a new collection of goal states for the next level of backchaining. This process is repeated until a knowledge state is constructed that includes the initial state of the system, or until there are no further knowledge states to be constructed.

Planning Randomized Strategies

The aim of this thesis is to analyze randomized strategies and explore methods for synthesizing these strategies. In the context of this thesis randomization takes the form of either guessing the current state of the system or of executing a randomizing motion. These two approaches are very similar, as is made clear by considering knowledge states. As an example, consider again the discrete representation for the peg-in-hole task of figure 2.2. Suppose that the initial knowledge state is $K = \{s_{\text{left}}, s_{\text{right}}\}$. This means that the system knows that it is on a horizontal

edge near the hole, but is unsure of which one. The state-guessing approach consists of randomly guessing that the actual state is either state s_{left} or state s_{right}, then executing a motion designed to attain the goal from that state. The randomizing-action approach consists of randomly moving either left or right, in the hope of attaining the goal. For this simple example the two approaches are trivially equivalent.

More generally, this example suggests that both state-guessing and action-randomization may be viewed as the random selection of a knowledge state that is a subset of the system's actual knowledge state at run-time. In other words, suppose that the system knows that it is located somewhere in the set K, and suppose further that this is not enough information to accomplish a task successfully. Then it makes sense to guess between some collection of smaller knowledge states $K_1, \ldots, K_q$ that cover K, assuming that for each of the knowledge states K_i there is a strategy for attaining the goal. Selecting one of the states K_i may be viewed either as guessing an artificial sensory interpretation set or as selecting a random sequence of actions. The sensory interpretation set is just the set K_i, while the sequence of actions is the plan associated with K_i for attaining the goal. This suggests that the synthesis of randomized strategies may be built on top of the backchaining approach used to synthesize guaranteed strategies. The guaranteed approach is simply augmented with an additional operator, SELECT, that permits the system to make random choices. Additionally, one must worry about whether it is possible to repeat this guessing operation should the first guess fail to attain the goal. Chapter 3 examines these issues in greater detail, while section 2.6 later in this chapter provides a further outline.

2.4 A Randomizing Example

Let us continue with an example. The purpose of this example is to demonstrate the relationship between guaranteed strategies, local progress, and randomization in a continuous space. The scene is the two dimensional plane. The state of the system is a point on this plane. The goal is a circle of radius r centered at the origin. The task consists of moving the system into the goal. This representation might, for instance, be the appropriate formulation of the problem of sliding a peg towards a hole on a level surface surrounding the hole. The point in this case corresponds to some reference point on the peg, while the plane corresponds to the two degrees of sliding freedom available to the peg.

If sensing and control are perfect, then the task is accomplished by sensing the start position, then moving in a straight line towards the origin, stopping once the circle is entered. Suppose however that sensing is imperfect. Then it may not always be clear in which direction to move. Let us look at a special case involving imperfect sensing, while retaining the assumption of perfect velocity control. In addition, we will assume that the goal is independently recognizable, that is, if ever the state of the system enters the goal, then some sensor will signal goal attainment. In the peg-

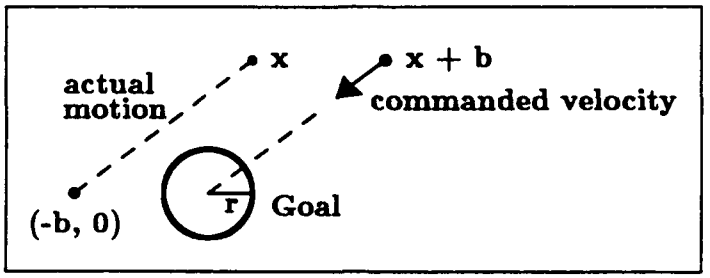


Figure 2.7: If there is a constant sensing bias and the system interprets the sensor as correct, then the system may converge to a point other than the goal.

in-hole example, this might be achieved by noting that the peg is falling into the hole, that is, by using force sensors to detect that contact with the surrounding surface has been broken. Another possibility is to sense the peg's height in the z-direction.

In general we will model sensing errors as error balls. Specifically, we will assume that if the actual location of the system is given by the point $\mathbf{x}$, then the sensor will return a sensed value $\mathbf{x}^* \in B_{\epsilon_s}(\mathbf{x})$, where $B_{\epsilon_s}(\mathbf{x})$ is the ball of radius ϵ_s centered at $\mathbf{x}$. As we have mentioned before, $B_{\epsilon_s}(\mathbf{x})$ represents the non-determinism in the system's knowledge of the sensor. It may be the case that all possible positions in $B_{\epsilon_s}(\mathbf{x})$ could be returned by the sensor, or simply that some subset could be returned. Further, the sensor may return values probabilistically distributed over $B_{\epsilon_s}(\mathbf{x})$, or it may return values in an adversarial manner. Without further information, the system must plan as if the sensor is actually acting as an adversary.

Suppose, however, for the sake of this example, that the sensor always returns the actual location of the system offset by a fixed bias $\mathbf{b}$. The actual bias is unknown to the system, merely its maximum magnitude $b_{\max}$ is known. So, one may take $\epsilon_s = b_{\max} = \max_{\mathbf{b}} |\mathbf{b}|$. In what follows we will draw all figures as if $\mathbf{b} = (b, 0)$, with $0 \leq b \leq b_{\max}$. However, this is just for convenience of exposition; the bias may lie anywhere inside the disk of radius $b_{\max}$.

Now consider what happens if the system continues to interpret the sensor as correct. See figure 2.7. If the system is at location $\mathbf{x}$, then the sensor will report that the system is at $\mathbf{x} + \mathbf{b}$. Aiming for the origin, the system thus will move in a straight line parallel to the vector $-(\mathbf{x} + \mathbf{b})$. This line points directly from the actual location $\mathbf{x}$ to the point $-\mathbf{b}$. If $b_{\max}$ is less than the radius of the goal, then the system will still successfully attain the goal. So suppose that the point $-\mathbf{b}$ lies outside of the goal. It is still possible for the system to wind up in the goal, namely if and only if the line connecting the two points $\mathbf{x}$ and $-\mathbf{b}$ passes through the goal circle of radius r (recall that there is no control error). See figure 2.8. Thus there is one region from which this strategy is guaranteed to attain the goal, and another from which this strategy causes the system to converge to the point $-\mathbf{b}$ (recall that the sensing error is a pure bias, without any superimposed noise). Of course, if $\mathbf{b}$ were known, then

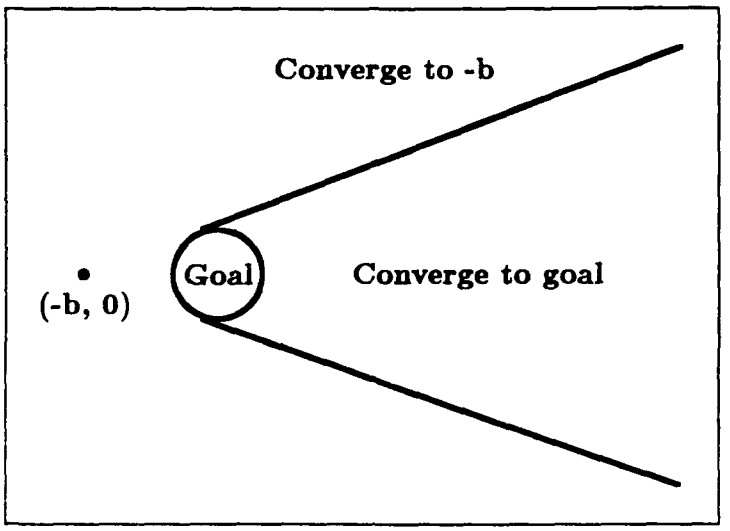


Figure 2.8: If the line from the system's starting configuration to the negative bias passes through the goal, then the system will converge to the goal. Otherwise, it will converge to the negative bias. This example assumes perfect velocity control.

the strategy could be modified to always achieve the goal, but **b** is unknown. Merely b_{max} is known to the system. Let us denote the region from which the strategy is guaranteed to attain the goal by P.

Suppose that we are interested in a simple feedback strategy designed to attain the goal, by making judicious use of sensors and randomizing when necessary. In particular, the strategy may not retain any past sensing information, but must base all its decisions on current sensed values. We will consider such a situation for the discrete case in section 3.12.3. In particular, we want a strategy that will make progress towards the goal when possible and otherwise will randomize its position. Consider then a circle of radius d, centered at the origin. The radius d is to be chosen in such a way that progress is possible towards the goal whenever a sensed value lies outside of the circle, while progress is not guaranteed whenever a sensed value lies inside the circle. We will discuss choosing d as a function of control and sensing uncertainty in greater detail in chapter 5. For the current example it makes sense to take $d = \epsilon_s = b_{max}$. This is because whenever a sensed value appears within ϵ_s of the origin, the system cannot be sure on which side of the origin the actual position is located, and thus cannot decrease the distance to the origin. It is true that the system can in general rule out locations that lie within the goal, and thus using $d = \epsilon_s$ is overly conservative if one is merely interested in making progress towards the goal, as opposed to making progress towards the origin. If one wanted to take this added information into account then using $d = \sqrt{\epsilon_s^2 - r^2}$ is appropriate (see figure 2.9). In either case, if a sensed value $\mathbf{x}^*$ appears outside of the circle of radius

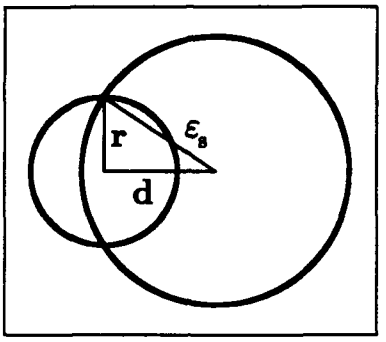


Figure 2.9: ϵ_s is the sensing uncertainty and r is the goal radius. d is the minimum distance form the origin that a sensed value must lie in order to guarantee progress towards the goal. If velocity control is perfect, taking $d = \epsilon_s$ is sufficient, but this figure shows that a smaller value of d is often possible.

d, then commanding a velocity in the direction $-\mathbf{x}^*$ is guaranteed to move all possible interpretations of $\mathbf{x}^*$, that is all points in the region $B_{\epsilon_s}(\mathbf{x}^*) - G$, closer towards the goal G. Furthermore, one can move in the direction $-\mathbf{x}^*$ for a total duration that changes distance by less than $2\,(|\mathbf{x}^*| - d)$, and still be sure that progress towards the goal has been made, independent of the actual location $\mathbf{x} \in B_{\epsilon_s}(\mathbf{x}^*) - G$.

Now consider shifting the circle of radius d by $-\mathbf{b}$. Denote the disk circumscribed by this circle by D. In the context of this special example, this disk represents the range of actual positions for which the returned sensor readings lie within distance d of the origin. Thus the disk consists of those locations of the system for which the simple feedback strategy cannot be sure of making progress towards the goal. (Recall, that the system knows $b_{\max}$ but not $\mathbf{b}$.) Observe, that if $d = \sqrt{\epsilon_s^2 - r^2}$ and $b = b_{\max} = \epsilon_s$, then D intersects the goal at the same points at which the boundary of the guaranteed region P intersects the goal circle. If d is larger than this, or b is smaller, then the disk D actually overlaps the region P. Thus there are three regions that characterize the behavior of this simple feedback strategy: (1) The region D, in which the strategy cannot guarantee progress, (2) the region P (or some subset thereof if D overlaps P) in which the simple feedback strategy can both guarantee progress and eventual goal convergence, and (3) the region $W = \Re^2 - (G \bigcup P \bigcup D)$, in which the strategy can guarantee progress locally but not eventual goal attainment. For this example, if the system starts off in W, then it will necessarily enter the disk D, simply because the system always moves towards the point $-\mathbf{b}$. See figure 2.10.

The region D correspond to a randomizing region. One possibility is for the system to randomly jump to some location whenever it finds itself unable to make progress, that is, whenever the sensor returns a value within distance d from the origin. Equivalently, the system could just move in a randomly chosen direction for some duration of time. These motions should be so chosen that there is a non-zero probability of entering either the region P or the goal G. For example, it may be

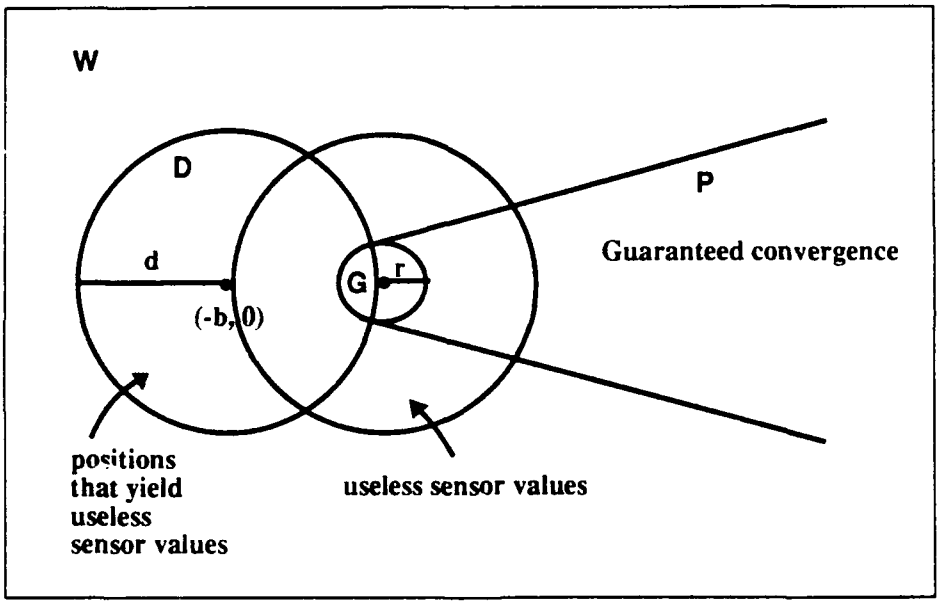


Figure 2.10: Range of positions and sensor values for which the system cannot decide in which direction to move. In the region D the system cannot make progress towards the goal. From the region P goal attainment is certain. From the region W progress is possible but not immediate goal attainment.

possible to randomly jump to some area A surrounding the goal, in which case the probability of entering the region P is just the ratio of the areas, that is, $|P \bigcap A|/|A|$. A typical execution trace of this strategy therefore consists of a series of straight-line motions into the randomizing disk, each of which is followed by a random motion out of the disk. Eventually one of these randomizing motions enters the preimage P, whereupon entry into the goal is guaranteed. The expected time until success is on the order of $|A|/|P \bigcap A|$ times the time required to execute a random motion. This time may be on the order of the diameter of A.

An alternative to using random jumps or extended random motions whenever a sensed value does not permit unambiguous progress towards the goal, is to execute a short random motion. The model is to employ a simple feedback loop in which all motions, both those executed deterministically and those executed randomly, are of a fixed short duration. This view of randomization follows the simple guessing strategy outlined in section 3.12.3. In the current context, the primitive actions are simply motion directions executed for some fixed small interval of time. Guessing between different knowledge states entails choosing a random motion direction. A simple feedback strategy that does not retain history thus does not have the capability of executing jumps or extended motions. Notice that this type of strategy has a considerably different behavior than the preceding one. In particular, if the system starts outside of the disk D, then it will head straight for the point $-\mathbf{b}$, either attaining the goal directly or entering the disk D. Once inside the disk D, the system will stray about randomly in that disk. Essentially, the boundary of the disk forms a barrier that is not crossed. This is because as soon as the system moves back out into region W, it will encounter a sensed value that permits progress towards the goal, thus sending the system right back into the disk. Thus, this strategy effectively amounts to a random walk inside the disk D. The random walk eventually crosses over into the goal G, whereupon the strategy terminates successfully. The expected time until success is on the order of the non-goal area inside the disk, that is $|D - G|$, times perhaps a logarithmic factor, depending on the location of the goal.[1]

We thus have two randomized strategies, of apparently different character. Certainly the random jumps appear to be of significantly different character than the short random motions. However, one can view a random jump as a strategy that randomly guesses the current state of the system then executes a motion designed to attain the goal assuming the guess is correct. Similarly, one can model the extended random motions as sequences of actions acting over short periods of time. The sequence may be viewed as the execution of a strategy with history, based on a randomly selected start region. In this manner, these randomizations fit nicely into the framework developed for the discrete case in chapter 3. In summary, one randomized strategy tries to escape the region D, in which sensing is useless, by randomly moving to a new start location, while the other strategy tries to escape this region by drifting across it towards the goal. The first may be viewed as randomization

[1] This is similar to the expected time of $n^2 \log n$ required to attain the origin on a two-dimensional $n \times n$ grid. See [Montroll].

with history, the second as randomization within a simple feedback loop.

Deciding which strategy to execute depends very much on the capabilities available to the system, as well as the expected times of success. For instance, if the preimage P is large compared to the area A into which the system jumps randomly and if the goal G area is small relative to the disk D, then it makes sense to randomize by jumping. Otherwise, it may make sense to randomize by performing a random walk.

An observation in favor of the random walk is the realization that for more general sensing and control uncertainties, there may not be a region P from which entry into the goal is guaranteed. In particular, the region of useless sensing may include the goal. This might happen if the actual bias has a magnitude considerably less than the maximum possible magnitude. In that case, even though the strategy can guarantee progress towards the goal whenever the system is far enough away, eventually, as the system approaches the goal, sensing becomes useless, and guaranteed progress must give way to random motions. In that case, both random jumps and random walks succeed only by actually attaining the goal.

What is interesting about this example is that both these randomized strategies succeed independent of the actual bias $\mathbf{b}$. In fact, the same strategies will succeed independent of the distribution of actual sensor values in the ball $B_{\epsilon_s}(\mathbf{x})$. The speed of convergence of course depends on the precise distribution but the existence of a solution does not. With slight modifications the strategies can be made to succeed in the presence of certain forms of control uncertainty as well.

This strategy is an example of the form to be discussed in section 3.12.4. In particular, the strategy takes advantage of the lack of an adversary who can forever keep the system from attaining the goal. This is evident in the assumption of a constant sensing bias. The bias plays the role of an unmodelled system parameter that cannot assume worst-case values at every location in state space. For the case $b = b_{\max}$ and $d = \sqrt{\epsilon_s^2 - r^2}$, this assumption ensures that for some approach direction there will be a guaranteed path to the goal. While this approach direction is not known to the system, the randomized motions ensure that it will be discovered eventually. More generally, there may not be a region of guaranteed success. In this case, the random walk ensures that the goal will be attained eventually. (N.B.: Implicit in this strategy is the assumption that there is no adversary who can bias the commanded motions sufficiently that they act in a non-random fashion, driving the system away from the goal.)

We will analyze the random-walk strategy again in chapter 5, and augment the strategy to account for control uncertainty. Further, assuming particularly nice distributions of sensing and control uncertainty, we will compute the expected progress at each point. The rest of this chapter will focus more on the manner in which both guaranteed and randomized strategies are computed in continuous cases. It is hoped that the example has provided a flavor of the approach.

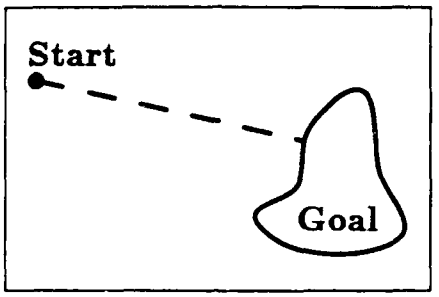


Figure 2.11: Given perfect sensing and control, a strategy for attaining the goal is simply a path to the goal.

2.5 Simple Feedback Loops

The main focus of this thesis is to develop an understanding of randomized strategies. This will be done both in the setting of full history and in the setting of simple feedback loops. Section 2.3.4 (page 73) explained the basic approach for planning randomized strategies that use full history, with further details appearing in chapter 3. This section is devoted to a quick overview of simple feedback loops with randomization. These were discussed in section 2.3.3. The region-attaining example of section 2.4 made use of a simple feedback loop. The basic structure of a simple feedback loop is well described by that example. In particular, a randomized simple feedback loop executes actions designed to make progress towards a goal when this is possible, and otherwise executes a random motion. The simple feedback loop only consults current sensed values in making its decisions. Again, chapters 3 and 5 examine feedback loops in greater detail.

2.5.1 Feedback and Uncertainty

Feedback in a Perfect World

The example of section 2.4 provided some of the motivation and the basic approach. Let us now develop these ideas slightly further, as a prelude to chapter 3. Consider first the setting of perfect control and perfect sensing. In such a perfect world a strategy for attaining a goal might consist of a series of paths that lead from any initial state to the goal. See figure 2.11. One might for instance take the paths to be the shortest paths to the goal. Sensing is not really required except perhaps to determine the starting location of the system.

Feedback with Imperfect Control

As one relaxes the assumption of perfect control, sensing becomes useful for correcting errors introduced during a motion. Again, a planner may specify a strategy that

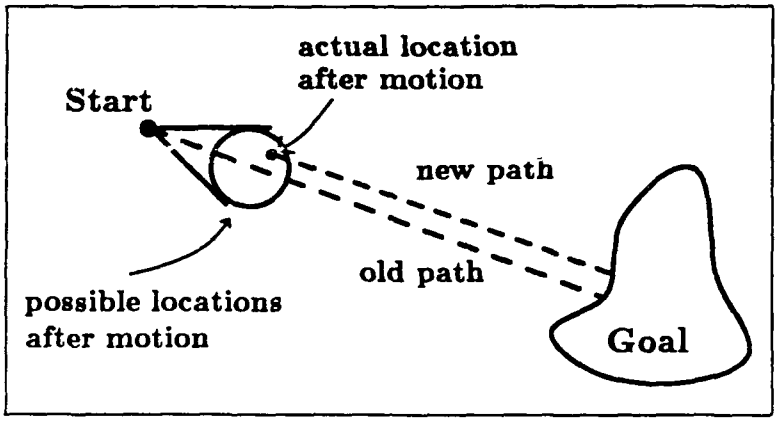


Figure 2.12: This figure shows a snapshot of a feedback strategy in which control is imperfect but sensing is perfect. At each instant the system determines a path to the goal from the current state.

consists of a collection of paths that lead to the goal. Sensing is used at run-time to determine which path the system is actually on at any instant. See figure 2.12. One now has a true feedback strategy. At each instant of time the sensed state of the system is used to decide on a proper course of action. The feedback strategy is a simple feedback strategy since it does not make use of past sensed values.

Observe that we have said nothing about how one actually comes by the paths that lead to the goal. In the perfect-world case these might come from a standard motion planner, or perhaps a shortest-path planner. In the perfect-sensing/imperfect-control world, one can use these same paths. In other words, the strategies determined for the perfect world may be used as nominal plans in the imperfect world. While it is true that one might be able to optimize the time to attain the goal by explicitly replanning, using for instance dynamic programming, this is not generally required merely to obtain a solution. Under simple bounds on the extent of the control uncertainty, and simple conditions on the paths, these nominal plans suffice to guarantee attainment of the goal. The conditions may be summarized by saying that the nominal paths should form a progress measure and that the control uncertainty should be small enough so that progress is possible at any state of the system. By a progress measure we essentially mean a scalar function that is continuous over the state space and that is reduced as one moves along any given path. Distance from the goal is one such measure. See also the work by [Khatib] on potential functions.

Feedback with Imperfect Control and Imperfect Sensing

Finally, let us relax the assumption of perfect sensing. We would like to extend the feedback approach outlined above. In particular, we would like to begin with a set of nominal paths or plans that lead from any location to the goal. The nominal paths

serve as a guide. At run-time the system repeatedly uses sensing to determine its actual location on one of these paths, thereby compensating for errors introduced by control uncertainty. This is a classic view of feedback. However, the presence of sensing uncertainty severely complicates the picture. The system now cannot ascertain precisely on which path it is located. Instead, there may be a collection of paths that are candidates for guiding the system to the goal. This collection is given by all paths that intersect the sensory interpretation set. So long as all these paths point in essentially the same direction, the system can find a motion direction which is guaranteed to make progress relative to the paths. However, it may easily be the case that some paths point in conflicting directions, so that the system cannot ensure that it will reduce its distance to the goal. This was the gist of the example of section 2.4. At this point randomization enters into the picture. If the system cannot guarantee progress relative to the nominal paths, then it should simply execute a randomizing motion. This ensures that there is at least a possibility of making progress, no matter where the actual location of the system is within the sensing uncertainty ball.

In short, we will think of a simple feedback loop as a feedback strategy that uses a progress measure to move towards the goal. The run-time knowledge state of the system is just its current sensory interpretation set. Whenever progress is possible for all states of the system within this knowledge state, the system executes a motion to make progress. Otherwise, the system executes a randomizing motion. Randomization is required to ensure ultimate goal attainment. This type of a randomized strategy is perhaps the simplest sensor-based strategy imaginable. It is a natural generalization of the feedback strategies used with perfect sensing. Strategies that employ history in making decisions are conceptually built on top of these simple strategies. In particular, randomization serves essentially the same role in all of these strategies, namely as a device to continue operation even when decisions cannot be made with certainty. It is merely that with the history-based strategies the effective state of the system is complicated by the influence of past information.

2.5.2 Progress in Feedback Loops

The Feedback Loop

The basic structure of a randomized simple feedback loop is given by the following pseudo-routine. The routine assumes that there is a non-negative scalar progress measure $\ell(x)$, defined at each point of the state space, that is zero at the goal. The function ℓ is often referred to as a *labelling* in the rest of the thesis. In general, additional conditions may need to be imposed on ℓ, such as continuity, and the absence of local minima. Recall also that $F_A(x)$ is the set of all states to which x might move under action A.

REPEAT until goal attainment:

Sense x^*.

Let $I(x^*)$ be the possible locations of the system.

FOR all actions A do:

For $x \in I(x^*)$, let $\Delta(x) = \max_{y \in F_A(x)} \{\ell(y)\} - \ell(x)$.

If $\Delta(x) < 0$ for all $x \in I(x^*)$,

then execute action A and exit from the **FOR** loop.

End_for

If no action A was executed,

then randomly select an action to execute.

End_repeat

Pseudo-code describing a simple feedback loop.

The inner **FOR** loop checks whether it is possible to make progress relative to the progress measure. If this is not possible, then a random action is executed. This feedback loop assumes that goal attainment is recognizable upon entry into the goal.

Velocity of Approach

The synthesis of these feedback loops is trivial assuming that a progress measure is given. Let us therefore turn to an analysis of such loops. The key issue is deciding how fast progress is made towards the goal. Thus it is useful to define the velocity of approach at each state of the system. Intuitively, we would like the velocity v_x to measure the rate at which progress is made whenever the system is in state x. We must be careful to define this quantity in a meaningful manner. The proper definition depends very much on the types of sensing uncertainty and control uncertainty that are in effect.

In a world with perfect control and perfect sensing, the velocity of approach is just the change in the progress measure, measured along the path to the goal. The velocity is negative whenever progress is being made. This velocity has a useful property. In particular, one can integrate the quantity $-1/v_x$ over a path to the goal in order to obtain the time required to attain the goal. This means that if for some v the velocity at each state x satisfies $v_x < v < 0$, then the time to attain the goal is bounded by $-d/v$, where d is the maximum starting distance from the goal. We would like our more general definition to possess this same property.

Much of the material in sections 3.4, 3.5, and 3.6 is concerned with defining velocity properly and establishing the bounding property just mentioned. There is a considerable difference between the probabilistic setting and the non-deterministic setting.

In the non-deterministic setting the natural definition of the velocity v_x is as the worst-case bound on the change in the progress measure whenever the system is in state x. In particular, the velocity at a state x is of the form:

$$v_x = \max_{\substack{\text{applicable} \\ \text{actions A}}} \max_{y \in F_A(x)} \{\ell(y) - \ell(x)\},$$

where ℓ is the progress measure as before. In order for this velocity to be negative, each of the terms inside the maximization must be negative. This says that the feedback loop is effectively a guaranteed strategy for attaining the goal. Given that the progress measure ℓ is based on a collection of nominal plans developed for a perfect world, one cannot actually expect that the velocities $\{v_x\}$ will all be negative. This suggests that the natural setting for simple feedback loops is in the probabilistic domain, rather that in the non-deterministic domain. Indeed in the probabilistic domain the definition of velocity leads to some interesting issues.

Random Walks

The natural domain for exploring simple feedback loops with probabilistic uncertainty is in the setting of Markov chains and their continuous counterparts. This is because for each state of the system, the simple feedback loop described above defines a range of probabilistic transitions. Each transition is the result of some action that the simple feedback loop might execute. An action is executed either as a result of obtaining a sensory value that permits making progress, or as a result of randomly selecting an action. Since sensing and control uncertainty are probabilistic, the net result is a set of probabilistic transitions.

As an example, consider again a two-dimensional peg-in-hole task for which the peg is in contact with a horizontal edge near the hole. Suppose that we have discretized the state space, as indicated in figure 2.13. In a perfect world, once the peg is in contact with a horizontal edge, a plan for attaining the goal consists of moving left if the peg is to the right of the hole, and moving right if the peg is to the left of the hole. There are thus two nominal paths for moving towards the goal. Said differently, a progress measure is given by the system's distance from the goal. Let us ignore the issue of control uncertainty and instead assume simply that the peg's motions consist of moving to neighbor states in the discrete representation of its state space. Now let us instantiate the simple feedback loop for this problem in the presence of sensing uncertainty. The feedback loop is based on the distance progress measure. [2]

[2] We should note in passing that the strategy is slightly silly, given the low-dimensionality of the state space. However, it is a convenient example for illustrating the construction and character of a simple feedback loop. A more complicated example was considered in section 2.4.

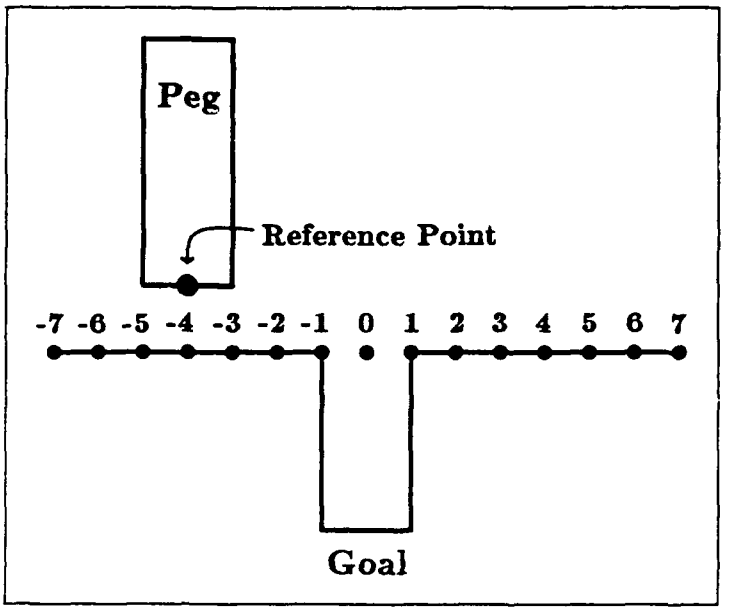


Figure 2.13: Discrete approximation of the horizontal state space of a peg-in-hole problem. State "0" corresponds to the goal.

1. Sense the current horizontal position.
2. Decide on a direction in which to move:
   (a) If the sensed value unambiguously determines the peg's position to be to the left of the hole, then decide to move right.
   (b) If the sensed value unambiguously determines the peg's position to be to the right of the hole, then decide to move left.
   (c) Otherwise, randomly pick left or right.
3. Move one step in the direction selected by the previous step, while simultaneously pushing down slightly.
4. Repeat steps 1 through 3 until the goal is achieved.

A simple feedback loop for inserting the peg of figure 2.13.

Let us analyze this strategy. Suppose that the sensor is symmetric. Then it suffices to consider the distance of the peg from the origin. Denote by a the distance of the peg's reference point from the origin. Let p_a be the probability that the sensor

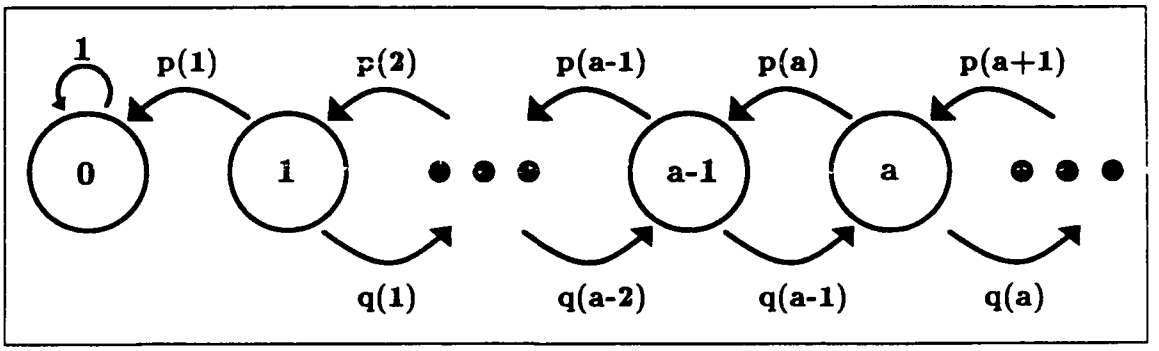


Figure 2.14: A Markov chain model for the discrete peg-in-hole problem of figure 2.13.

will return an unambiguous reading when the peg is located at distance a from the hole. By an unambiguous sensor reading we mean a sensed value x^* all of whose interpretations $I(x^*)$ lie either completely to the left or completely to the right of the hole. Then the probability of moving towards the hole is

$$p_{hole}(a) = p_a + \frac{1}{2}(1 - p_a) = \frac{1}{2} + \frac{1}{2}p_a.$$

Figure 2.14 shows the resulting system, modelled as a simple Markov chain. [Here $p(i)$ is shorthand for $p_{hole}(i)$, and $q(i) = 1 - p(i)$.]

The precisc value of p_a and thus of $p_{hole}(a)$ depends on the sensor, of course. Observe, however, that $p_{hole}(a) > 1/2$ whenever $p_a > 0$. In short, there is a natural drift towards the origin. Indeed, the expected change in the distance from the origin is given by:

$$\begin{aligned} \Delta a &= (-1)\,p_{hole}(a) + (+1)\,(1 - p_{hole}(a)) \\ &= -2\,p_{hole}(a) + 1 \\ &= -p_a. \end{aligned}$$

In other words, on average, the system decreases its distance from the goal by p_a per step. It thus makes sense to define the velocity at the point a to be $v_a = -p_a$.

We see in this example one of the key issues that arises in the analysis of randomized strategies, in particular, of simple feedback loops. This is the question of whether sensing is strong enough to pull the system towards the goal on average. In this one-dimensional example we see that the natural drift is indeed towards the goal everywhere. In more complicated spaces this need not always be the case. Part of chapter 5 is devoted towards analyzing one such example, based on the two-dimensional problem of section 2.4. We will see that for nicely behaved sensing

and velocity errors, there is an unbounded annulus about the origin within which the system moves towards the origin on average. However, once the system lies within a certain distance of the origin, the sensing information becomes less useful. Instead, the randomizing actions tend to push the system outward. Although eventually the system will approach arbitrarily closely to the origin, the natural drift is away from the origin on the average. This places a lower bound on the size of the goal region required to ensure fast convergence.

More generally, one can define the expected velocity at a state to be the expected change in the progress measure. A considerable portion of chapter 3 is devoted to proving that this definition of velocity in the probabilistic setting has many of the same properties as does the usual notion of velocity in a deterministic world. In particular if the expected velocity at every state is bounded from above by some number $v < 0$, then the expected time to attain the goal is bounded from above by $-d/v$, where d is the maximum starting distance from the goal.

An attractive aspect of the probabilistic definition of velocity is that it captures the notion of progress on the average. In order to converge to a goal rapidly a strategy thus need not make progress at every instant in time, so long as it makes progress on the average. This is a considerably more flexible definition than what is available in a non-deterministic world. This is because in a non-deterministic world all constraints are formulated in terms of worst-case behavior. One desirable trait of randomization in general is that it permits one to mix the notions of worst-case and average-case behaviors. Thus even in an adversarial world one can sometimes gain an advantage by purposefully randomizing one's actions. This is the idea put forth in section 2.4. Even though one may not be able to ensure progress on any given attempt, by randomizing one can at least ensure progress eventually, and in some cases, one can ensure progress on the average.

2.6 Strategies Revisited

We saw in section 2.2 that there are essentially four dimensions that define the types of tasks that arise in robot motion planning with uncertainty. It is easy to confuse the methods for these different problems, so let us recall the four dimensions briefly.

One dimension corresponds to the level of uncertainty in the actions. The categories of action uncertainty that we discussed were: DETERMINISTIC, PROBABILISTIC, PARTIALLY ADVERSARIAL, and NON-DETERMINISTIC. A second dimension corresponds to the level of uncertainty in sensing. The categories of sensing uncertainty that we discussed were: PERFECT, PROBABILISTIC, PARTIALLY ADVERSARIAL, NON-DETERMINISTIC, NEARLY-SENSORLESS, and SENSORLESS. A third dimension corresponds to the type of strategy used to solve the task. The two categories that we discussed were GUARANTEED and RANDOMIZED. Finally, the fourth dimension corresponds to the amount of history used by these strategies in making their decisions. The two extremes that we discussed were given by FULL HISTORY and SIMPLE FEEDBACK. In some sense there is a fifth dimension,

corresponding to the type of state space, but we will ignore this dimension in the current categorization since most of the results generalize from the discrete case to the continuous setting.

Focusing for the moment on the two dimensions of strategy type and history usage, the following table describes the contribution of this thesis.

		Strategy Type	
		Guaranteed	Randomized
History	None	**LMT**	Thesis
	Full	**LMT; DP**	Thesis

Focus of the thesis.

The entry "**LMT**; **DP**" refers to the work by [LMT] on preimages and the general dynamic programming approach for planning guaranteed or optimal strategies. See chapter 4 for a discussion of preimages in the continuous domain and chapter 3 for a discussion of dynamic programming in the discrete domain.

This thesis does not discuss much the synthesis of guaranteed strategies that use no history. In general, simple feedback loops are best thought of in the probabilistic or randomized domains, since they are generally not guaranteed to converge in a predetermined number of steps. However, some work has been done in this area in the context of robot motion planning. Clearly, guaranteed strategies that use no history may be viewed as a special case of preimage planning [LMT]. Other special cases and extensions are discussed in [Erd84], [Buc], and [Don89], among others.

Turning to the dimensions of control and sensing uncertainty, the following table describes the the types of tasks considered either directly or indirectly by this thesis. Essentially, the natural approach is to pair up non-deterministic control with non-deterministic sensing, and probabilistic control with probabilistic sensing. Entries with a "$\surd$" refer to task specifications that are special cases of either the general preimage framework or the material discussed in this thesis. Entries that specify section or chapter numbers refer to material treated in detail in the thesis.

		Control (Action) Uncertainty		
		Perfect	Probabilistic	Non-Deterministic
Sensing Uncertainty	Perfect	$\surd$	$\surd$	$\surd$
	Probabilistic	$\surd$	§3.4; §3.5; §5	—
	Partially Adversarial	§2.4	§5	§2.4 ; §3.12.4
	Non-Deterministic	$\surd$	—	§3.6–§3.11; §4
	Near-Sensorless	$\surd$	$\surd$	§3.13
	Sensorless	$\surd$	$\surd$	§3.13

Descriptions of tasks considered by this thesis.

One issue that these tables do not highlight is the relationship of non-deterministic models to probabilistic models. In some cases the world may behave probabilistically,

even though the model is non-deterministic. Section 5.2 treats this topic briefly. The topic arises naturally in the analysis of strategies formulated in terms of the non-deterministic model. A guaranteed or randomized strategy that assumes a non-deterministic description of uncertainty is certain to succeed independent of the actual instantiation of errors. However, in order to perform a specific rather than a worst-case performance analysis, it is often useful to assume a particular instantiation of the sensing and control errors, such as assuming some probabilistic model. For those cases it is important to understand the relationship between the worst-case model and the probabilistic model. Indeed, most of chapter 5 is concerned with the analysis of a simple randomized strategy, modelled after the example of section 2.4. The strategy is general enough to succeed under a variety of worst-case scenarios. In order to gain some appreciation for the behavior of the strategy, however, it is useful to assume a pair of idealized probabilistic distributions describing the sensing and control errors.

2.7 Summary

This chapter has briefly outlined the basic focus of the thesis. The chapter defined different types of uncertainty, and different approaches for planning strategies that solve tasks in the presence of uncertainty. The focus of the thesis is on randomized strategies, with a particular emphasis on simple feedback loops. A simple feedback loop only considers current sensory information in deciding on a course of action. Randomized simple feedback loops expect as input a progress measure, perhaps in the form of a nominal plan for attaining the goal. The randomized feedback loop attempts at each instant to move in a manner that makes progress. If this is not possible, then the system makes a random motion. The chapter included an example consisting of a randomized strategy for pushing a peg on a surface into a two-dimensional hole.

More generally, randomization is useful because it permits solutions to tasks for which there are no guaranteed solutions, because it simplifies the planning process, and because it reduces brittleness. Brittleness is reduced because randomization can blur the significance of environmental details. Rather than requiring a detailed analysis of an environment, a system can instead rely on randomization to effectively ignore details below a certain scale.

Chapter 3

Randomization in Discrete Spaces

This chapter examines the role of randomized strategies in the solution of tasks that may be represented by a set of discrete states and actions. The chapter will also indicate how to plan strategies, with an emphasis on finding strategies that may be planned and executed quickly. In particular, it will be shown that there are some tasks for which randomized solutions execute more quickly on the average than do guaranteed solutions in the worst case. In general, of course, a given task may not have a guaranteed solution, but we will see that under very simple conditions there is always a randomized solution to a task specified on a discrete space. However, the expected execution time may be very high.

3.1 Chapter Overview

This first section provides a brief guide to the organization of this chapter.

Basic Definitions

The first main section (§3.2) presents a more detailed version of the basic definitions of chapter 2, specialized to tasks on discrete spaces. The section begins with the definition of tasks in the non-deterministic setting, then moves on to the probabilistic domain. Next the section considers the problem of planning guaranteed or optimal strategies in the probabilistic setting. In particular, the *Dynamic Programming Approach* is reviewed. This planning approach applies with slight variations to the non-deterministic setting as well. Finally, the section ends with some technical subsections that elaborate on the definition of knowledge states and a connectivity assumption. Knowledge states reflect the uncertainty with which a system knows its location at run-time. The connectivity assumption rules out consideration of tasks in which massive failure can occur.

Random Walks

As we noted in chapter 2, random walks form one of the most basic type of randomized strategies. In particular, the results developed in the context of random walks are basic to the understanding of simple feedback loops. Section 3.4 considers random walks, and section 3.5 introduces the notion of expected progress. This second section defines the expected velocity at a state relative to a labelling of the state space. The section proceeds to show that this notion of an expected velocity possesses some of the standard properties of a deterministic velocity. In particular, if the expected velocity at all states points towards the goal and is uniformly bounded away from zero, then an upper bound for the time to attain the goal is given by the distance from the goal divided by the velocity bound.

Planning with Randomization

Sections 3.6 through 3.11 consider the general problem of planning strategies that purposefully randomize. This planning approach is built on the dynamic programming approach used for generating guaranteed strategies.

Extensions and Specializations

The remaining sections discuss various extensions and specializations of randomized strategies. Of particular interest are near-sensorless tasks. In these tasks the system must rely almost entirely on its predictive ability to attain a goal. The only sensing information available is whether or not the goal has been attained. By including this one bit of information it is possible to develop randomized strategies structured as loops that repeatedly attempt to attain the goal.

3.2 Basic Definitions

This section presents the basic definitions of actions, sensors, and tasks on discrete spaces. Section 2.2 already explained some of these concepts. The current section elaborates on more of the technical details. The presentation of these definitions is in the context of both non-deterministic and probabilistic actions and sensors. The basic approach is the same for both types of uncertainty. Subtle differences between the non-deterministic and probabilistic cases are mentioned as necessary.

3.2.1 Discrete Tasks

We should convince ourselves that there are tasks that may be represented in discrete terms. Recall that some examples were given in chapter 2, in particular in section 2.2.1. A typical such task is given by the stable configurations under gravity of a polyhedral object resting on a planar surface. Indeed, if one drops a polyhedral object onto a horizontal table under the influence of gravity, with probability one it

will come to rest on one of the faces comprising the convex hull of the object. There are finitely many such faces. Thus, although the natural configuration space of the object is a six-dimensional space consisting of the three translational and three rotational degrees of freedom of the object, if the task only requires examination of the object's stable resting configurations, then the induced state space is finite. Determining the transitions between these stable states may require a dynamical analysis in the full six-dimensional (or higher) state space of the object, but once that analysis has been performed, the planning of operations can occur in the finite and discrete state space.

Even though the state space may be discrete it may not be immediately apparent that the set of transitions between the states is finite. Although there actually may be a continuum of actions, im many cases there is a natural partitioning of this continuum into a finite collection of equivalence classes, where each action in an equivalence class has the same effect in terms of the transitions on the underlying state space. For instance, if we are interested in the stable resting configurations of an object on a table, we may alter those resting configurations by exerting a force on the object through its center of mass. In that case, we can partition the space of forces into regions whose qualitative behavior differs across regions but is identical within a region. For instance, forces that point into the friction cone, thus causing no motion, constitute one region. Other regions might include those forces that cause sliding, and those that cause the object to flip from one stable configuration to one or more other stable configurations.

The representation of tasks is a difficult issue. In some cases, problems that appear to reside in a continuum state space, may be transformed into equivalent or similar problems that reside in finite state spaces. The details of the transformation tend to be task-specific, although often stability under some set of actions may be used as a criterion in defining the discrete states. The work of Brost ([Brost85] and [Brost86]) involves such a transformation for the problem of pushing and grasping planar polygonal objects. Mani and Wilson [MW] used a similar transformation in their work on pushing, and Erdmann and Mason [EM] employed a stable-under-gravity transformation in their work on orienting planar parts in a tray.

A slightly different type of transformation is given by the examples of gear-meshing and object-sieving cited in Chapter 1. Here in some sense there are two states, namely SUCCESS and FAILURE. A complicated higher-dimensional analysis was used to determine the effect of a particular action, that is, to compute the probability of success in each example. However, once that probability had been computed, the task could be represented by a discrete state space, with a probabilistic transition graph. Certainly, more complex graphs can be envisioned, especially for the sieve-task, in which one could imagine a series of sieves arranged vertically above each other. In that case a natural discrete graph is given by states corresponding to the regions between the different sieve levels. Assuming that one does indeed randomize the object's configuration between sieves, there is no need to accurately model this configuration, and it becomes sufficient to collapse all configurations between two sieves into a single state. Of course, if one is interested in synthesizing strategies by varying the possible motions through the sieves, then one may have to return to the

full two-dimensional continuum configuration space of the part being moved. Again, this may not be such a problem, if one decides to limit the possible sets of motions to a finite class, either by only considering finitely many or by partitioning them into equivalence classes relative to some relation.

3.2.2 Discrete Representation

This section provides the formal representation of tasks in which the relevant state space and action set are discrete and finite. The development will assume non-deterministic actions and sensors. More specialized actions and sensors, such as probabilistic ones, are discussed in chapter 2. Additionally, sections 3.2.3 and 3.2.4 discuss probabilistic actions, sensing, and planning.

States

In a discrete problem we are given a finite set of states $\mathcal{S} = \{s_0, s_1, s_2, \cdots, s_n\}$, and a finite set of actions $\mathcal{A} = \{A_1, A_2, \cdots, A_m\}$. In principle, one could define several sets of actions, each set representing the actions that are applicable at a particular state. However, we will simply assume that every action is applicable at every state. This is an unrestrictive assumption that simplifies the notation in discussing the effects of actions when the current state is unknown.

Actions

The actions are non-deterministic, that is, given some starting state s, the result of applying an action A may be any one of a possible set of states $F_A(s) = \{s_{i_1}, s_{i_2}, \cdots, s_{i_k}\} \subseteq \mathcal{S}$. This set is called the *forward projection* of the state s under action A. Figure 3.1 shows how we will represent non-deterministic actions graphically. In the figure, action A_1 may have one of three results when applied to state s_0, but has precisely one result when applied to states s_1, s_2, or s_3. Symbolically, we would write this as:

$$\begin{aligned} A_1: \quad & s_0 \mapsto s_1, s_2, s_3 \\ & s_1 \mapsto s_1 \\ & s_2 \mapsto s_2 \\ & s_3 \mapsto s_3. \end{aligned}$$

Section 2.2.2 contained some examples of non-deterministic actions.

As another example of a non-deterministic action, consider an Allen wrench in contact with a tabletop, as shown in the top portion of figure 3.2. Suppose a force is applied through the center of mass as shown. Depending upon the coefficient of friction, the accuracy of the applied force, the position of the center of mass, and so forth, there are two possible final stable states of the Allen wrench. These are shown

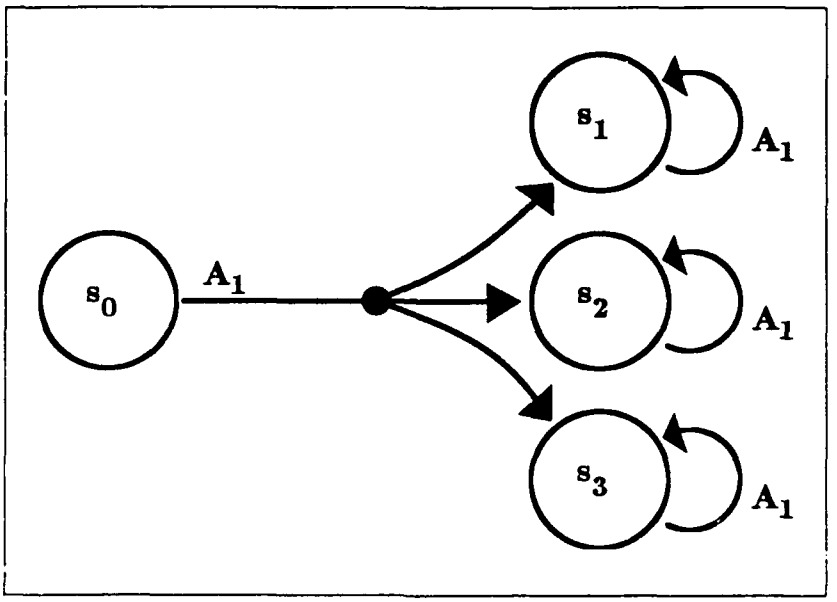


Figure 3.1: Graphical representation of a non-deterministic action A_1.

in .ower portion of figure 3.2. If the parameters determining the motion of the wrench cannot be modelled accurately, for instance if the coefficient of friction is unknown, then the action should be modelled non-deterministically.

Tasks

We will assume that tasks are specified as goal states that should be achieved. That is, there is some set $\mathcal{G} \subset \mathcal{S}$ of states, whose attainment constitutes completion of the task. By attainment, we will mean *recognizable attainment*, that is, the system is in a goal state and knows that it is in a goal state.

Similarly, the system is assumed to initially start in some subset $\mathcal{I} \subseteq \mathcal{S}$ of states.

Sensors

Finally, we should comment on sensors. Sensors may or may not be available. We shall model a sensor as a relation between states and subsets of states. In other words, given that the system is in some state, the sensor returns some subset of possible interpretations. See section 2.2.3 for a description of possible types of sensors and sensory interpretations. In general, the sensor need not be deterministic, that is, for a given state, the sensor may return one of several possible sets of interpretations. However, we will assume that there exists at least one possible interpretation set for any given state. This assumption is always easily satisfied, since one can if necessary take this interpretation set to be the entire state space. See also section 3.2.5 below.

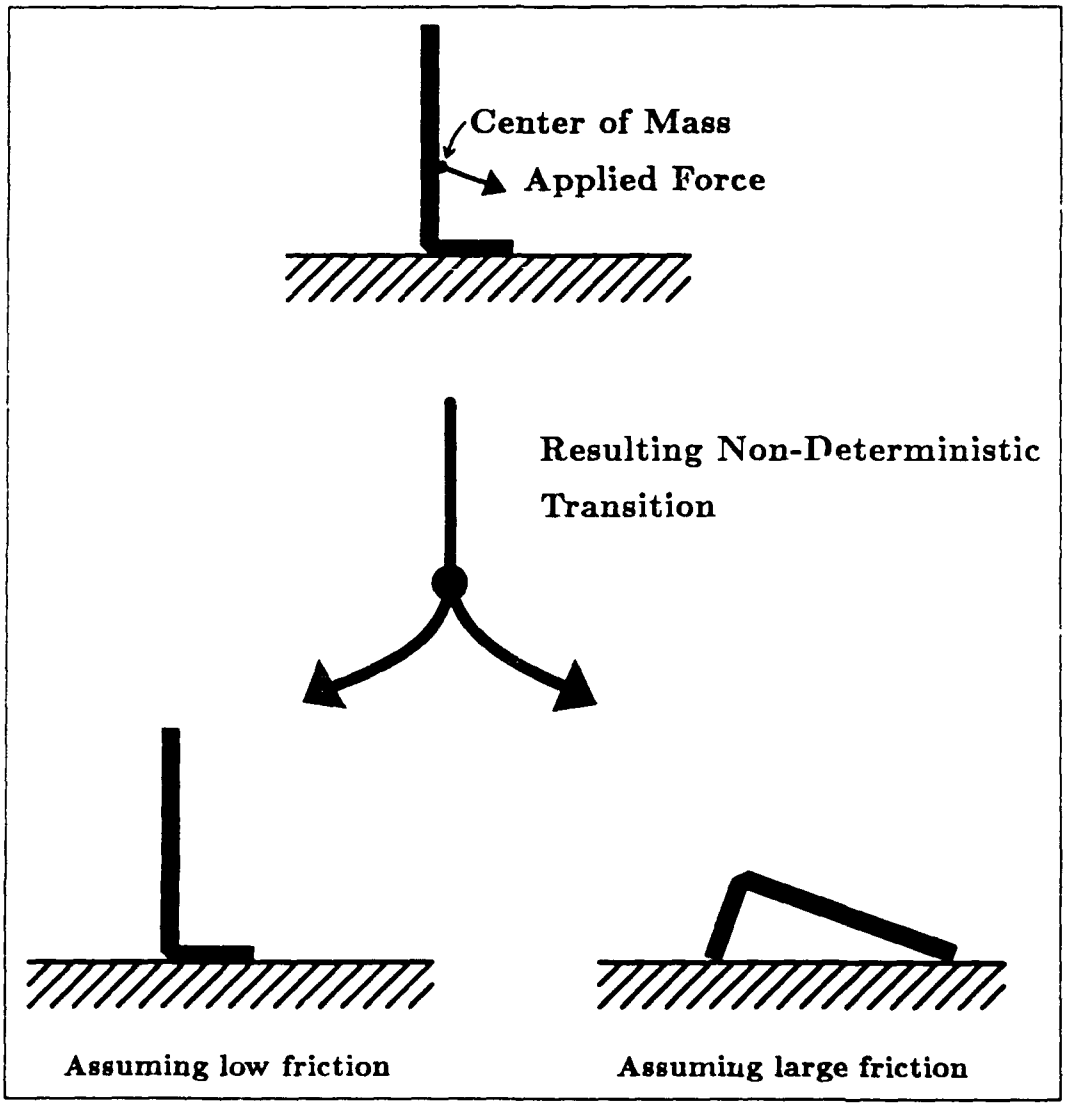


Figure 3.2: The force applied to the Allen wrench in the top of the figure will cause the wrench either to slide without rotation or to rotate and possibly slide. The actual motion depends on the coefficient of friction. If the coefficient of friction is not known it is useful to model the force as a non-deterministic action.

Functional Representation

If we wanted to express actions and sensors as functions, then the proper encoding would be:

$$\text{If } A \in \mathcal{A}, \text{then } A : \mathcal{S} \rightarrow 2^{\mathcal{S}},$$

where $\mathcal{S}$ is the set of states and $2^{\mathcal{S}}$ is the set of all subsets of $\mathcal{S}$. Similarly, we can model the sensor function as a mapping Ξ from states to all sets of subsets of states:

$$\Xi : \mathcal{S} \rightarrow 2^{2^{\mathcal{S}}}. \tag{3.1}$$

In other words, for any state s, $\Xi(s)$ is a collection of sets, say $\Xi(s) = \{I_1, \cdots, I_\ell\}$. We will refer and have been referring to each I_i as a *sensory interpretation set*. $\Xi(s)$ describes all possible sensory interpretation sets that might arise at run-time whenever the system is in state s. This means that at run-time the physical sensor can return some value whose interpretation is one of the subsets I_i of the state space.

For a perfect sensor, the sensing function becomes $\Xi(s) = \{\{s\}\}$, for every $s \in \mathcal{S}$. Abusing notation we will sometimes write this as $\Xi(s) = s$. On the other extreme, if no sensor is available, then Ξ reduces to $\Xi(s) = \{\mathcal{S}\}$ for every $s \in \mathcal{S}$. Again, abusing notation we will sometimes write this as $\Xi(s) = \mathcal{S}$. See section 2.2.3 for some examples of sensing uncertainty.

These representations are intended only to describe the character of the range of actions and sensors. In other words, actions map to sets of states, while sensors can return any one of a collection of sets of states. Particularly in the case of sensors the representation (3.1) is much too general. We will impose additional constraints on this representation in order to derive a physically reasonable model in our discussion on knowledge states below (see section 3.2.5).

Two comments should be made. First, sometimes it is useful to break the sensor function into two parts. The first part models the sensor values that may result upon examination of the sensor when the system is in a particular state. We will denote a possible such sensor value by s^*. The second part models the interpretations of these sensor values as sets of states. In particular, if s^* is an observed sensor value, then $I(s^*)$ will denote its set of interpretations. See, for instance, [TMG]. Often the second of these functions follows from the first, so we have decided to collapse the representation. However, for some of the examples in the thesis, when we derive the sensory interpretation sets possible at a given state, we may first determine the actual values $\{s^*\}$ returned by a physical sensor, then map these to their interpretation sets $\{I(s^*)\}$. In any event, no serious information is lost by mapping directly from states to possible interpretation sets.

The second comment concerns the domain of the sensor function, which was taken to be the state space. Sometimes a sensor's value may depend on a sequence of states, or on some other parameter, rather than on just the current state. This is particularly true in the continuous time case, where a physical sensor may be averaging noisy measurements over time before reporting these to the control system. In the discrete

case this seems less likely, and so is not modelled here. Further, any dependence on an unmodelled parameter can always be collapsed into further non-determinism in the function Ξ. This conservatively preserves the sensor's response, although it may weaken the power of the executive system in making decisions. Another approach is to augment the definition of the system's state in order to incorporate the sensor state.

Notice that further variations on this model are possible. For instance, the effect of actions could be made time-dependent, as could the results returned by the sensors. We will not consider such variations.

3.2.3 Markov Decision Processes

Non-Determinism and Knowledge Paucity

We have thus far chosen to represent transitions as non-deterministic transitions. This reflects the presence of uncertainty in the actions we are modelling. This model does not incorporate any further knowledge about the nature of the uncertainty in the actions.

In some cases the uncertainty may be due to a paucity of knowledge in modelling the actions on the state space, rather than an inherent non-determinism in the actions themselves. For instance, it may turn out in figure 3.1 that action A_1 actually always moves from state s_0 to state s_1, but this is simply not known to the task-system.

Probabilistic Actions and Optimality

In other situations one may have enough information to think of the transitions between states as being probabilistic. In other words, associated with each action and each start state is a distribution function, describing the probabilities of attaining the states in the forward projection F_A of the start state. If actions may be described using probabilistic transitions, then it is natural to formulate optimality problems in terms of expected cost for some cost function defined on the states and the transitions between them. A typical problem is to find the sequence of actions that attains a goal state in minimum expected time. Such problems are known as *Markov Decision Processes*, and have been studied for several decades. Recent results by [Pap] and [PT] have characterized these problem in terms of PSPACE. In particular, the general problem of finding the minimum expected cost sequence of actions of a given length is shown to be PSPACE-hard. Various specializations of the problem are actually in PSPACE. Of particular interest are the perfect-sensing and no-sensing cases. The latter problem is shown to be NP complete, while the former is shown to be P-complete. A standard approach for computing optimal decisions in the perfect-sensing case is to use dynamic programming (see, for example, [Bert]).

Probabilistic Sensors

Note that sensing may also be formulated probabilistically. There are at least two natural ways of doing this. In our current representation, if the system is in state s, then $\Xi(s)$ is a collection of sets. This means that at execution time the sensor can return any one of the sets in $\Xi(s)$. Each set is a sensor interpretation describing the possible states of the system. One possibility for a probabilistic sensor would be to define a probability distribution over this collection of sets. In other words, for each state of the system, the sensor has a certain probability of returning any given set of interpretations. Another possibility is to not model Ξ as returning different possible sets of states, but to instead model Ξ as returning different possible probability distributions. In other words, for each state of the system s, $\Xi(s)$ is a collection of probability distributions over the state space. One can merge these two variations by assigning probabilities to each of the probability distributions in the collection $\Xi(s)$. Indeed, this is often the approach taken. For instance, if we have a Gaussian sensor, then, for each state of the system, we can associate a probability density to the possible sensor values. And by inverting these distributions using a Bayesian approach, we can think of each sensor value as defining a probability distribution on the state space. See also section 3.2.6.

3.2.4 Dynamic Programming Example

This section reviews and demonstrates the use of dynamic programming by a simple example. The main reason for reviewing dynamic programming is its use of backchaining, a method that is useful for computing guaranteed plans in the presence of uncertainty. We will state the example in a probabilistic setting. However, it should be understood that the same approach applies to planning guaranteed strategies in the presence of non-deterministic uncertainty. We briefly indicate the planning process in the non-deterministic case.

A Probabilistic Example

The example consists of a series of states connected by actions that have probabilistic transitions (see figure 3.3). After any transition, sensors report the resulting state with complete accuracy. The starting state can also be sensed with perfect accuracy. The task is to determine a mapping from knowledge states to actions that maximizes the probability of attaining the goal in a specific number of steps. This mapping constitutes a plan or a strategy for attaining the goal. Knowledge states are discussed further in sections 2.3.3 and 3.2.5. Intuitively, a knowledge state describes the system's current best estimate of a region in which it is located. A knowledge state is determined by current and past sensory information, as well as by predictions based on executed actions. With perfect sensing, the relevant knowledge states are simply the actual states of the system.

The basic idea of dynamic programming is to maximize (or minimize) some value function in terms of the actions available and the number of steps remaining to be

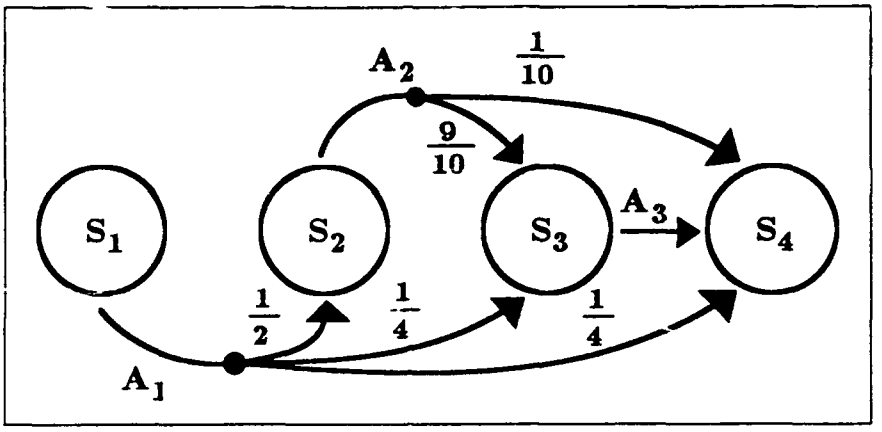


Figure 3.3: A state graph with probabilistic transitions. There are four states and three actions. The label on an arc indicates the probability that the transition will be taken when the specified action is executed. All transitions not indicated are self-transitions.

executed. At each stage, an action is selected for each state that would maximize the value function given that there remain a certain number of steps to be executed. This maximization is performed by first recursively determining the maximum values obtainable for each state given one fewer step, then selecting an action for the current state that maximizes the expected value of moving to another state. One starts the whole process off by assigning values to each state that reflect the value of the value function if no actions whatsoever remain to be executed. This is exactly what it means to backchain from a goal. For the situation in which one is looking for strategies with maximal probability of success, the value function represents the probability of achieving the goal in the remaining steps. Goal states are initially assigned a value of 1; non-goal states a value of 0. Further, the value in the k^{th} stage of the computation for a particular state is the probability of attaining the goal from that state in at most k steps, assuming that the system can sense perfectly and that it always executes the maximizing action at each state.

The backchaining maximization of dynamic programming may be depicted by a table (see below). The columns of the table correspond to the stages in the backchaining process; the rows correspond to the knowledge states of the execution system. Counting from right to left, an entry in the k^{th} column of the table for knowledge state K_i specifies the action to be taken at run time if there remain k time-steps in which to execute actions and if the system's current knowledge state is K_i. The entry in the table might also specify the value of the value function computed at that point in the backchaining process. For instance, the entry might specify the maximal probability of success given that there remain k actions to execute and given that the system is in some state s_i.

Consider now figure 3.3, which depicts four states and three actions. The transitions resulting from the execution of actions are labelled with probabilities. All actions are applicable in all states. However, for simplicity, we have not drawn transitions that leave a state unchanged. For instance, if the current state is state s_2, then action A_2 moves to state s_4 with probability 1/10, while action A_1 remains in state s_2 with probability 1.

Suppose that state s_4 is the goal state. Then the value assigned to the four states at the zeroth stage of backchaining is 0 for states s_1, s_2, and s_3, and 1 for state s_4. At the first stage, the values assigned are 1/4 for state s_1, 1/10 for state s_2, and 1 for states s_3 and s_4. These values reflect the maximum probabilities of attaining the goal in one or zero steps.

The following table reflects the computations for four stages. The entries in the table are the computed maximum probabilities, along with the correct action to take in that state, given the number of steps remaining. In this example, the optimal actions for each of the states happen to be the same across stages, but that need not be the case in general.

Steps Remaining				
3	2	1	0	
1; A_1	11/20; A_1	1/4; A_1	0	s_1
1; A_2	1; A_2	1/10; A_2	0	s_2
1; A_3	1; A_3	1; A_3	0	s_3
1; stop	1; stop	1; stop	1; stop	s_4

States

Probabilities of success: Optimal actions.

The table shows that the goal can be achieved with certainty from any state using no more than three steps, as one would expect.

Complexity

Computing such a table out to k stages for a state space with n states and $O(m)$ actions can be done straightforwardly in time $O(k\,m\,n^2)$. In particular, the solution is in P (polynomial time). [In this complexity estimate we are ignoring the precision of the transition probabilities, that is, we are assuming that addition and multiplication can be done in constant time.]

A Non-Deterministic Example

For completeness of exposition suppose that the transition graph of figure 3.3 is non-deterministic rather than probabilistic. In this case the value function to be maximized by the dynamic programming approach is a boolean function. A "1" of this function corresponds to guaranteed success, while a "0" corresponds to possible failure. The dynamic programming table for the non-deterministic case is almost identical in appearance to the table for the probabilistic case. A blank entry in the

table indicates that success cannot be guaranteed in the number of steps remaining from that state. In other words, the boolean value function has value "0". Conversely, an entry with an action A_i indicates that the boolean value function has value "1", that is, eventual goal attainment is guaranteed if the system executes action A_i.

Again, the table shows that the goal can be achieved with certainty in at most three steps.

Steps Remaining					
3	2	1	0		
A_1				s_1	
A_2	A_2			s_2	States
A_3	A_3	A_3		s_3	
stop	stop	stop	stop	s_4	

Actions that guarantee goal attainment.

3.2.5 Knowledge States in the Non-Deterministic Setting

This and the next section explain how to represent the possible states of a system at execution time, that is, what the executive's knowledge states are. A planner must of course reason about more knowledge states than actually occur during execution, since in general at planning time the outcome of a sensing operation will not be known precisely.

Forward Projection

First, let us look at the case in which actions are non-deterministic and sensors return possible sets of interpretations. In this case, at any given time during execution the actual state of the system is known only to be one of possibly many. Thus the space of knowledge states is simply the set of all subsets of the state space, namely 2^S. Given a set K_1 of possible states that the system could be in, and an action A, the result of executing action A is a new knowledge state K_2, given by:

$$K_2 = \bigcup_{s \in K_1} F_A(s).$$

In other words, K_2 is the union of all the possible non-deterministic transitions resulting from possible states in K_1. Notice that this knowledge is equivalent both at execution time and at planning time. The process of forming K_2 is called *forward projecting* set K_1 under action A, and is written $K_2 = F_A(K_1)$.

Forward projections possess a nice property. The forward projection of a collection of sets is just the union of the forward projections of the individual sets. This is summarized in the following lemma.

Lemma 3.1 *Let $\{K_i\}$ be a collection of knowledge states, and let A be a non-deterministic action. Then*

$$F_A\left(\bigcup_i K_i\right) = \bigcup_i F_A(K_i).$$

Proof. Clear from the definition. ∎

Sensing

Let us now turn to the procedure by which a run-time executive might update its knowledge state using sensing. Given a knowledge state K_1 and a sensory interpretation set I, the resulting knowledge state is $K_2 = K_1 \bigcap I$. For sensing, however, knowledge at execution time can be considerably different than at planning time: at execution time the set I is known, whereas at planning time the system only knows that I will come from one of several possible sets of interpretations.

See again figure 2.6 on page 71, which shows the process of forward projecting a knowledge state and intersecting the forward projection with the current sensory interpretation set.

The analogue to the distributive property of forward projections is given for sensory interpretation sets by the distributive property of set intersections.

Lemma 3.2 *Let $\{K_i\}$ be a collection of knowledge states, and let I be some sensory interpretation set. Then*

$$\left(\bigcup_i K_i\right) \bigcap I = \bigcup_i (K_i \bigcap I).$$

Proof. Clear. ∎

In the next few paragraphs we will augment the process by which a system updates its knowledge state using sensory information. Indeed it is sometimes useful to make use of more structure than that provided simply by intersecting the current knowledge state with the current sensory interpretation set.

Constraints on Sensors

We will make one further set of assumptions concerning the possible sensory interpretations. The purpose of these assumptions is to rule out inconsistencies that would be possible given the unrestrictive definition of the sensing function Ξ in equation (3.1).

Consider figure 3.4. The figure shows the system's current knowledge state K which includes the actual state of the system x. The sensed value is x^*, and the sensory interpretation set $I(x^*)$ is given by a disk centered at x^*. Unfortunately this disk does not overlap the knowledge state. Thus if the system updates its knowledge state by computing $K \bigcap I(x^*)$, the result will be the empty set. The problem here is that the sensory interpretation set does not include the actual state of the system.

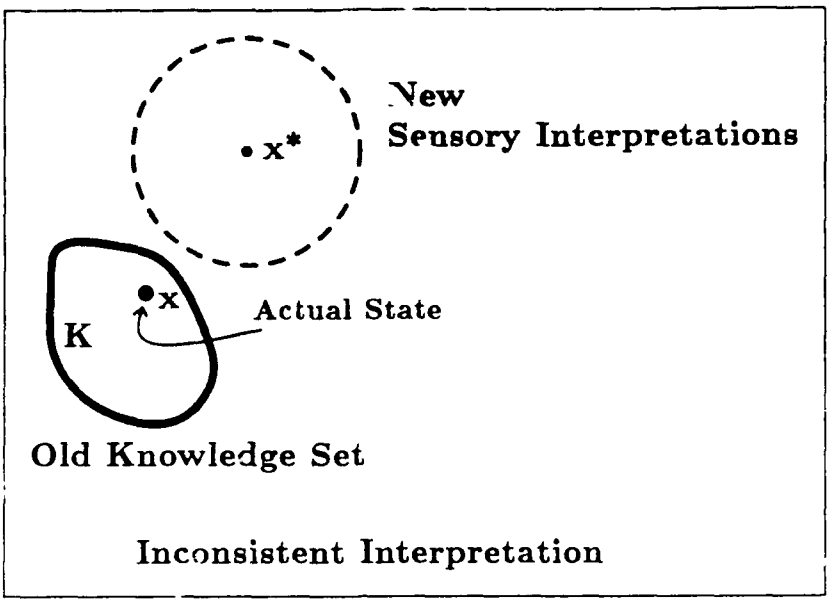


Figure 3.4: The sensory interpretation set in this figure does not overlap the system's previous knowledge state. This implies an inconsistency. The actual state of the system is x.

This leads to our first restriction on the definition of the sensing function Ξ. We require that a sensory interpretation set always include the actual state of the system.[1]

Partial Sensing Consistency Requirement. Let s be a system state, and let $I \in \Xi(s)$ be a possible sensory interpretation set returned by the sensor when the system is in state s. We require that $s \in I$. This means simply that a state is always an interpretation of any sensor value to which it can give rise.

Inconsistent Knowledge States

The example of figure 3.4 introduced the notion of a sensory interpretation set that is inconsistent with the current knowledge state. For the example of the figure, the inconsistency is removed by the partial sensing consistency requirement. This is because the knowledge state K contains the actual state of the system. However, if the run-time executive's knowledge state does not contain the system's actual state, then it is still possible to obtain a sensory interpretation set that does not overlap

[1] In the probabilistic case, it is sometimes useful to relax this requirement. In particular, when sensory interpretations are density functions with infinite tails it is useful to insist merely that the sensory interpretation set cover the actual state of the the system with some sufficiently high probability. We will make use tacitly of this version of the partial sensing consistency requirement in chapter 5. See in particular section 5.2.

the executive's run-time knowledge state.

There is a subtle issue here that requires further explanation. In particular, why should a system's knowledge state not contain the actual state of the system? This may seem peculiar, since the knowledge state is intended to reflect the certainty with which the system knows its actual state. If the knowledge state does not contain the system's actual state then something must be wrong in the modelling of the information available to the system, either in the modelling of the actions or in the modelling of the sensors. This means that if ever the system encounters the empty set upon having updated its knowledge state, then the system knows immediately that something is wrong in the modelling of the task. In turn, this suggests that we need not worry about inconsistent interpretations, since if an inconsistency ever does occur, it must be due to an unmodelled parameter, that is, an event beyond the scope of the task description. The smart thing to do is to stop the task execution and to try to model the unknown parameter.

This explanation is correct, but it ignores part of the motivation for the thesis. In particular, we would like to develop methods for solving tasks without having full knowledge of all the parameters in the system. The particular approach taken in this thesis is to actively randomize, either by guessing sensor values or by executing random actions. The randomization is intended to blur the significance of these unmodelled parameters. Formally, as we indicated in section 2.3.4 on page 73, one view of randomization is as the random guessing of possible knowledge states. In other words, the actual knowledge state of the system is too large for it to execute a useful strategy, so the system simply guesses that its actual state lies in a smaller set. The smaller set is then assumed to be the knowledge state. Actions and sensing update this smaller knowledge set, rather than some larger knowledge state, as if it were the correct description of the run-time executive's certainty. This approach will be further explored starting in section 3.9.

We see then that the set $K \bigcap I(x^*)$ readily can be empty, where $I(x^*)$ is some sensory interpretation set and K is a knowledge state. This is because the knowledge state K may have been randomly selected during a guessing step of a randomized strategy. This is actually very useful. For, if the run-time executive ever observes that $K \bigcap I(x^*) = \emptyset$, then it knows that the actual state of the system cannot be in K. This implies that the original guess of K as an appropriate knowledge state must have been wrong. Having determined that the guess was wrong, the system can then guess again, or try some other strategy.

*Interpreting Sensors More Carefully

This section may be skipped on a first reading. It deals with a technical point regarding the consistency of the sensing function Ξ.

Thus far we have only imposed one restriction on the character of sensory interpretations, namely the partial sensing consistency requirement. This restriction merely insured consistency between the actual state of the system and observed sensory interpretations. The requirement may be interpreted as ensuring that sensory

interpretation sets are not too small. However, thus far we have not imposed a constraint in the other direction, to ensure that sensory interpretation sets are not too large.

If sensory interpretation sets are larger than necessary, then it may be to a strategy's advantage to perform a more complicated operation than merely intersecting the sensory interpretation set with the current knowledge state. The next few paragraphs indicate what is meant by a sensory interpretation set that is too large and how a system can better update its knowledge state. Fortunately, it is possible simply to modify the sensory interpretation sets prior to execution time so that they are not too large. This modification will be formulated in terms of a second consistency requirement.

A fairly natural way in which sensory interpretation sets may be too large is if they are chosen conservatively to bound the actual state of the system. For instance, consider figure 3.5.

This is an example on a continuous space, but the moral of the example applies equally well of course to discrete spaces. In this two-dimensional example the sensing error ball has a radius varying as a function of the system's x-coordinate. In particular, if the actual state of the system is (x, y), then the range of possible sensor values is given by a circle centered at (x, y) with radius $x/4$. This example is supposed to abstract the notion of a position-dependent error function. Suppose that the work space is given by the square $[0, 1] \times [0, 1]$. If (x^*, y^*) is an observed sensor value, then one may take as the sensory interpretation set $I(x^*, y^*)$ the circle of radius $1/4$ centered at (x^*, y^*). Clearly this interpretation set is too large for small values of x, but it is definitely a conservative approximation, and satisfies the partial sensing consistency requirement.

Now consider the example of figure 3.6. There are two knowledge states, given by the two vertical strips $K_1 = \{(x, y) \mid 0 \leq x \leq 0.4\}$ and $K_2 = \{(x, y) \mid 0.7 \leq x \leq 1\}$. Let the observed sensor value be $(x^*, y^*) = (0.6, 0.5)$, with corresponding sensory interpretation set $I(x^*, y^*) = B_{1/4}(0.6, 0.5)$. Clearly this sensory interpretation set overlaps each of the knowledge states K_1 and K_2. If a system simply intersects sensory interpretation sets with knowledge states, then the system would conclude that its location could be either in the set K_1 or the set K_2. On the other hand it is clear to us as outside observers that no point in K_1 could have given rise to the sensor value $(0.6, 0.5)$. This is because the maximum range of possible sensor values for a point $(x, y) \in K_1$ is a disk of radius 0.1. This means that the maximum possible x^*-value observable if the system is in K_1 is 0.5. Only system states in the set K_2 could give rise to the observed sensor value $(0.6, 0.5)$. However, again, not all of the system states in the intersection $K_2 \bigcap I(x^*, y^*)$ could give rise to the observed sensor value (x^*, y^*). In short, even the intersection of the sensory interpretation set with K_2 is an overestimate.

The previous example is not surprising. After all, having conservatively bounded the actual sensory interpretation sets, one would expect that the run-time knowledge states computed by the system might overestimate uncertainty. The question is whether the structure of the function Ξ is internally consistent (see definition below).

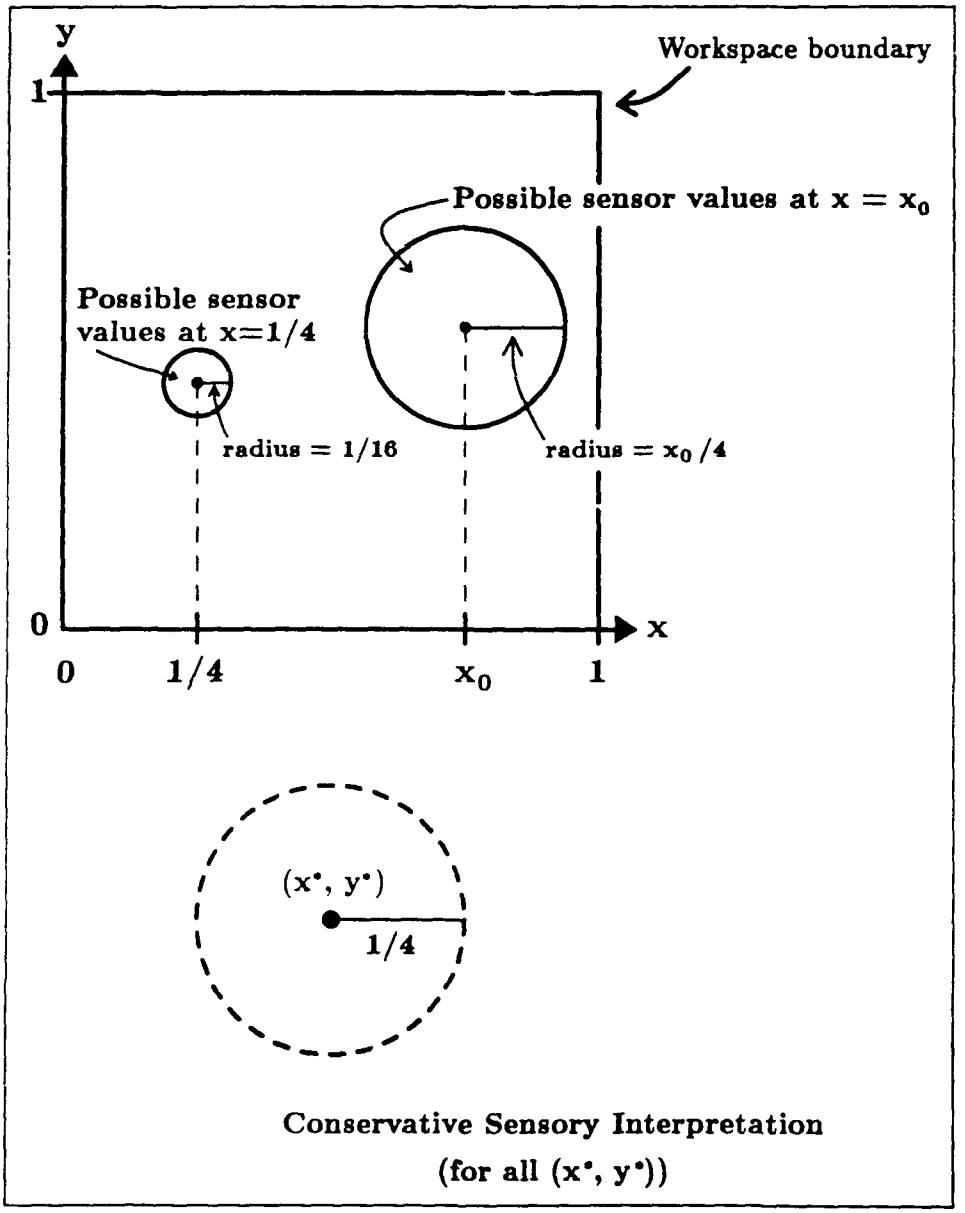


Figure 3.5: The sensing error ball in this example is position dependent. If the actual state of the system is (x, y), the possible sensor values are given by a ball of radius $x/4$ centered at (x, y). Over the indicated workspace a conservative approximation to the sensory interpretation set for an observed sensory value (x^*, y^*) is given by a ball of radius 1/4.

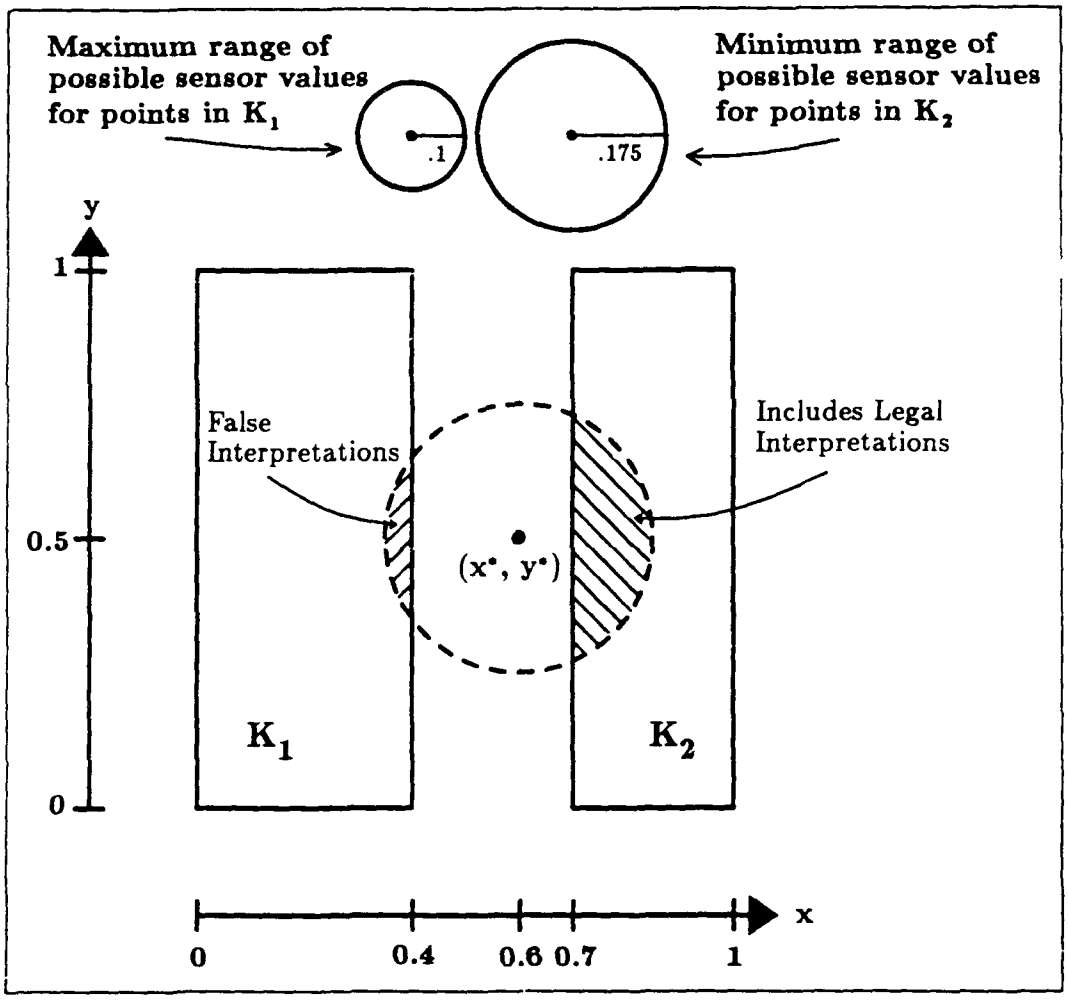


Figure 3.6: The sensing error ball about the sensed value $(x^*, y^*) = (0.6, 0.5)$ overlaps both knowledge sets K_1 and K_2. However, the observed sensor value can only correspond to an actual system state in the set K_2. This is because the range of sensor values for points in K_1 has a maximum radius of 0.1. The position-dependent possible sensor values are described in figure 3.5.

In particular, let us consider the collection of interpretation sets $\Xi(s)$ for some state $s = (x, y)$. This collection might be of the form

$$\Xi(x,y) = \bigcup_{|(x^*,y^*)-(x,y)|<x/4} \{B_{1/4}(x^*, y^*)\}, \tag{3.2}$$

or it might be of the form

$$\Xi(x,y) = \bigcup_{|(x^*,y^*)-(x,y)|<1/4} \{B_{1/4}(x^*, y^*)\}, \tag{3.3}$$

to name two extremes. The first collection (3.2) consists of all balls of radius 1/4 whose centers (x^*, y^*) lie within distance $x/4$ of the actual state of the system. In other words, this collection correctly models the actual sensor values that the system might observe, but then conservatively bounds the interpretations of these sensor values. The second collection (3.3) consists of all balls of radius 1/4 whose centers (x^*, y^*) lie within distance 1/4 of the actual state of the system. In other words, this collection not only conservatively bounds the interpretations, but also conservatively assumes that a greater range of sensor values is possible than the system will actually observe.

If the sensor function Ξ is of the form given by (3.2), then the system can obtain additional information by investigating the function Ξ that it cannot obtain simply by intersecting sensory interpretation sets with knowledge states. In particular, if the sensing function is of the form (3.2), then the system can rule out interpretations in the set K_1 of figure 3.6, while retaining some or all of the interpretations in the set K_2. On the other hand, if the sensing function Ξ is of the form given by (3.3), then the system can do no better than to intersect sensory interpretation sets with knowledge states.

Definition. In some sense the sensing function given by (3.3) is *internally consistent*. By this we mean that the system cannot gain any extra information by explicitly examining the structure of the sensing function, as by examining the collections $\Xi(s)$ for all states s. Instead, all the information upon observing a given sensor value s^* is available in the interpretation set $I(s^*)$.

In contrast, the sensing function given by (3.2) is not internally consistent. There are two basic ways to make this function internally consistent. One is to modify the collections $\Xi(s)$ so that they conservatively bound the range of possible sensor values as in (3.3). The other is to modify the actual interpretation sets so that they are exact rather than conservative bounds.

One question of interest is how a system should update its knowledge state if the sensing function Ξ is not necessarily internally consistent. Suppose, in particular, that the system's current knowledge state is K_1 and that it has observed a sensor value with interpretation set I. Let us define an operation $\bigcap'$ that updates the knowledge state K_1 using both the sensory interpretation set I and information about the structure of the function Ξ. We want the updated knowledge state K_2 to consist of all states in

both K_1 and I that could have given rise to the sensory interpretation set I. Formally, $K_2 = K \bigcap' I$, where

$$K \bigcap' I = \{ s \in K \bigcap I \mid I \in \Xi(s) \}. \tag{3.4}$$

This expression provides a formula for ensuring that the sensing function Ξ is internally consistent. Expression (3.4) says that one should delete from a sensory interpretation set I any states that could not possibly give rise to I.

We can summarize the condition that a sensing function be internally consistent by imposing an additional consistency requirement. The purpose of this requirement is to capture the condition under which the operator $\bigcap'$ reduces to the operator $\bigcap$. Combining this condition with the partial sensing consistency requirement yields the following consistency requirement.

Full Sensing Consistency Requirement. Let Ξ be a sensing function on a state space $\mathcal{S}$. Denote by $\Xi(\mathcal{S})$ the set of all possible sensory interpretation sets, that is, $\Xi(\mathcal{S}) = \bigcup_{s \in \mathcal{S}} \Xi(s)$. We say that *a sensing function satisfies the full sensing consistency requirement* if the following condition holds for all states $s \in \mathcal{S}$:

$$I \in \Xi(s) \text{ if and only if } s \in I \text{ and } I \in \Xi(\mathcal{S}).$$

In other words, if a state can give rise to a sensory interpretation set then that interpretation set must include the state itself, and conversely. It was the converse requirement that was missing in the example of figure 3.6. It makes a lot of sense to impose the partial sensing consistency requirement, as sensors that do not satisfy it do not seem very useful. Once one has the partial consistency requirement, it is easy enough to impose the full consistency requirement. After all, suppose that one sees a sensory interpretation set I which nominally contains the state s. If one examines $\Xi(s)$ and discovers that $I \notin \Xi(s)$ then one knows that s could not possibly have given rise to I. Thus one may as well replace I with $I - \{s\}$. This was the gist of the operation $\bigcap'$ defined by (3.4) above.

For the sake of completeness we prove the following lemma, which establishes that $\bigcap$ and $\bigcap'$ really are the same operator when the full sensing consistency requirement holds.

Lemma 3.3 *Suppose Ξ is a sensing function on a state space $\mathcal{S}$ that satisfies the full sensing consistency requirement. Then $\bigcap' = \bigcap$.*

Proof. Let $K \subseteq \mathcal{S}$ be a knowledge state, and let $I \in \Xi(\mathcal{S})$ be a sensory interpretation set. We need to show that $K \bigcap I = K \bigcap' I$. By the definition (3.4), we see that $K \bigcap' I \subseteq K \bigcap I$. Thus we need only to establish the reverse inclusion. Suppose that $s \in K \bigcap I$. In particular $s \in I$. By the full sensing consistency requirement it follows that $I \in \Xi(s)$. The definition (3.4) then establishes that $s \in K \bigcap' I$, as desired. ∎

To summarize, the full sensing consistency requirement ensures that the sensory interpretation sets are neither too small nor too large. This means that all the information available to the system from the sensing function is contained in the individual sensory interpretation sets. This is clearly a desirable property. In particular, it permits the system to update knowledge states with sensory information using set intersections.

3.2.6 Knowledge States in the Probabilistic Setting

Next, let us consider the case in which all actions are probabilistic and in which sensory interpretations are also probabilistic. Specifically, let us assume that for a given action A, if the system is in state s_i then it will move to state s_j with probability p_{ij}. The matrix (p_{ij}) is known as a *probability transition matrix.* Similarly, a sensory interpretation is really a *conditional distribution vector* (ι_j). This says that if the system was thought to be in state s_j with probability p_j before the sensory operation, then after the sensory operation it is thought to be in state s_j with probability $p_j\,\iota_j/\iota$, where ι is a normalization factor required to ensure that the resulting probabilities form a true distribution (see below). [As this expression indicates, the numbers $\{\iota_j\}$ are usually determined by a Bayesian analysis of how different states can give rise to different sensor readings.]

In the probabilistic setting, the state of the system is known with some probability. Thus the natural knowledge states are probability distributions over the state space $\mathcal{S}$. In other words, a knowledge state is a collection of $|\mathcal{S}|$ non-negative numbers that add up to one. If the current knowledge state is $K_1 = \{p_0, p_1, \cdots, p_n\}$, and action A has probability transition matrix (p_{ij}), then the effect of applying action A is a new knowledge state $K_2 = \{q_0, q_1, \cdots, q_n\}$, where

$$q_i = \sum_{j=0}^{n} p_j\, p_{ji}.$$

This is just a probabilistic forward projection. The sum is similar to a union operation; it measures the probability of moving to state s_i from each state s_j in the system, multiplied by the probability of having actually started in that state.

As we have already indicated, a sensory interpretation I corresponding to some observed sensor value s^* is of the form

$$I = (\iota_0, \iota_1, \ldots, \iota_n).$$

Here ι_j is the conditional probability of observing s^* given that the state of the system is s_j. The sensory interpretation I changes a knowledge state from $K_1 = \{p_0, p_1, \cdots, p_n\}$ to $K_2 = \{p_0\,\iota_0/\iota, p_1\,\iota_1/\iota, \cdots, p_n\,\iota_n/\iota\}$. This is just the probabilistic equivalent of set intersections in the non-deterministic case. Note that

$$\iota = \sum_{j=0}^{n} p_j\,\iota_j.$$

3.2.7 Connectivity Assumption

We would like to make a connectivity assumption that ensures that the goal is reachable from each possible state of the system. In the probabilistic setting this assumption amounts to the condition that for each start state there is a sequence of arcs with non-zero transition probabilities that attains the goal. In the non-deterministic case the assumption amounts to the condition that even in a worst-case scenario there is always some sequence of arcs that leads from each state to the goal.

The purpose of this connectivity assumption is to rule out massive disasters from which recovery is impossible. In other words, there are no non-goal trap states or trap subsets. An example of a trap is a snap-fit. Other examples include orienting parts over a deep lake or walking in a tiger-filled jungle. Generally, in the domain of tasks involving the manipulation or assembly of rigid objects, the connectivity assumption will be met so long as one can apply arbitrary forces and torques on the objects being manipulated.

The reason for ruling out such massive failures is to prevent randomized strategies from failing irrecoverably. In a more general setting in which certain parts of the state space must be avoided at all costs, one must restrict randomization to the safe part of the state space. If this is not possible, then randomization should not be applied.

Probabilistic Setting

In the case that actions are specified as probabilistic transitions, the connectivity assumption amounts to verifying that the transitive closure of each state in the induced transition graph contains a goal state. The transitive closure of a state in a directed graph is the set of all states reachable from that state by some path. By the induced transition graph we mean the directed graph whose vertex set is the set S of underlying states, and whose directed arcs are given by the set of all transition arcs whose associated probabilities are non-zero. This set is computed by considering the set $\mathcal{A}$ of all possible actions.

Non-Deterministic Setting

In the situation that actions are specified as non-deterministic transitions, we need a stronger condition than for the probabilistic case. In the probabilistic case we essentially verify the possibility of moving from any state to a goal by looking for some sequence of transitions connecting the state to the goal. Since each arc has a positive probability of being executed, the sequence as a whole has positive probability of being executed, so it is possible to reach the goal from the given state. In the non-deterministic case, such a test is not sufficient. This is because some arcs appear in the diagram simply due to a paucity of knowledge in modelling the underlying physical process. There is no guarantee that the arcs will ever be traversed. [See also the section on adversaries (§1.3)].

In order to understand the difference between the non-deterministic and the probabilistic case consider figure 3.7. In both Part A and Part B, if one interprets

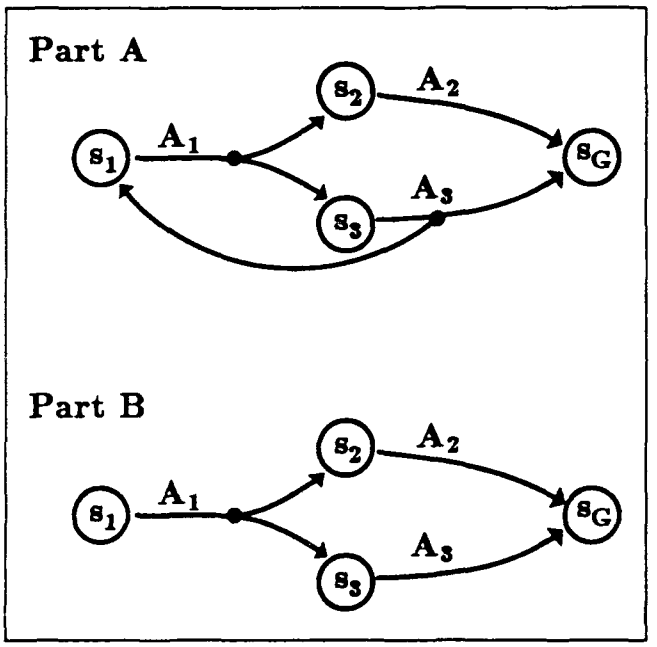


Figure 3.7: Both tasks satisfy the connectivity assumption whenever the arcs have positive probabilities of being executed. However, only the task of Part B satisfies the connectivity assumption if the arcs are interpreted as worst-case non-deterministic transitions.

all the arcs as probabilistic arcs with positive transition probabilities then the task satisfies the connectivity assumption. In other words, from any state there is a sequence of transitions that attains the goal with non-zero probability. However, if we interpret the arcs as worst-case transitions, then only the task of Part B satisfies the connectivity assumption. In Part A, from a worst-case point of view, there is a possibility that the system will forever loop between states s_1 and s_3.

Let us formalize the connectivity assumption. As we stated on page 112, even in the worst case there should for each state exist a sequence of actions that leads to the goal. Recall that a non-deterministic action A can cause a given state s to transit non-deterministically to any one of a set of states $\{s_1, \cdots, s_k\}$. There is no further information in the model, and one must thus be prepared that any one of the transitions can occur. That is what is meant by a worst-case model. We will refer to an *instantiation* of such a non-deterministic transition as a particular choice s_i. In other words, on a particular execution of action A while the system is in state s, the result is *instantiated* as state s_i. By an *instantiation of all possible actions* we mean a choice s_i for all actions $A \in \mathcal{A}$ at all possible states $s \in \mathcal{S}$. An instantiation of all possible actions yields a directed graph whose vertex set is $\mathcal{S}$ and whose arcs are the directed arcs defined by the instantiation. We will refer to a particular such graph as an *instantiated transition graph*. Figure 3.8 shows the four instantiated transition graphs that are possible by instantiating in all possible ways the non-deterministic actions of the graph in Part A of figure 3.7. Notice that for one of the graphs, two states are disconnected from the goal. This says that in a worst-case scenario it might not be possible to reach the goal. As we shall see, this also says that there is no perfect-sensing strategy for attaining the goal from an arbitrary state.

Definition. We will say that it is *certainly possible* to reach a set of goal states $\mathcal{G}$ from a given state s if for any instantiated transition graph there is some path that leads from the state s to some goal state in $\mathcal{G}$.

This definition captures the notion that no matter how the world behaves within the non-determinism allowed by the specified actions, there is some path for attaining the goal. The definition says nothing about whether the system can actually compute that path or execute it. After all, the system is not necessarily aware of the actual instantiations of the actions it executes. The connectivity assumption merely says that it is "certainly possible" to attain the goal, that is, that no adversary can prevent it for certain.

Looking ahead slightly, this connectivity assumption facilitates the use of randomized strategies. This is because a system can randomly guess what the instantiated graph looks like. Having made its guess, the system can execute a sequence of actions that follows a path to the goal. If the system guessed correctly, then these actions attain the goal. Otherwise, the system fails to attain the goal, but can try again. The connectivity assumption ensures that on each guess there is a non-zero probability of guessing correctly, uniformly bounded away from zero. Thus, eventually, the system will guess correctly.

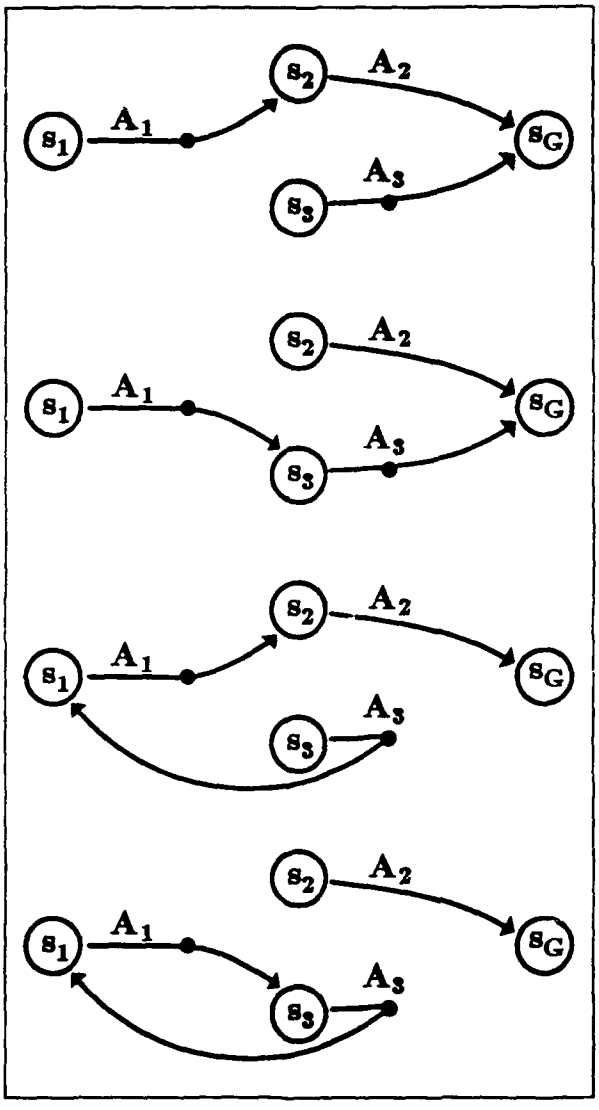


Figure 3.8: These four instantiated transition graphs describe the different possible worst-case scenarios for the task of Part A of figure 3.7. The absence of a path to the goal for states s_1 and s_3 in the fourth graph indicates that it is not "certainly possible" to reach the goal from an arbitrary state in the state space.

In fact, it turns out that the connectivity assumption in the non-deterministic setting is equivalent to the existence of a guaranteed perfect-sensing strategy. This is proved below. Furthermore, the perfect-sensing strategy need not have more steps than there are states.

Connectivity Tests

Thus in both cases we have a simple test for verifying goal connectivity. In the probabilistic case the test involves computing transitive closures. In the non-deterministic case the test consists of searching for a perfect-sensing strategy. This may be done quickly, using dynamic programming, as explained in section 3.2.4. Notice that the probabilistic test need not yield an optimal strategy, and the non-deterministic test need not yield a guaranteed strategy for an arbitrary sensing function. The probabilistic test merely yields **some** strategy, while the non-deterministic test yields a guaranteed strategy **given a perfect sensor**. The tests, and hence the assumption, are definitely weaker than the general planning problem itself.

Goal Reachability and Perfect Sensing

And now the two claims. The first establishes the equivalence between goal reachability and perfect-sensing strategies, the second shows that a guaranteed strategy under perfect sensing requires few steps.

Claim 3.4 *Let $(\mathcal{S}, \mathcal{A}, \Xi, \mathcal{G})$ be a discrete planning problem, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, Ξ is the sensing function, and $\mathcal{G}$ is the set of goal states.*

It is "certainly possible" to reach $\mathcal{G}$ from any state $s \in \mathcal{S}$ if and only if there exists a guaranteed perfect-sensing strategy for attaining $\mathcal{G}$ from any state $s \in \mathcal{S}$.

Proof. First, suppose that there exists a perfect-sensing strategy that is guaranteed to move the system from any state s to some goal state. Then for any instantiated transition graph there must be a path from s to $\mathcal{G}$. This path may be determined by executing the perfect-sensing strategy while selecting action transitions as prescribed by the instantiated transition graph.

Conversely, suppose that for any instantiated transition graph and any state $s \in \mathcal{S}$ there is a path from s to $\mathcal{G}$. We would like to exhibit a perfect-sensing strategy for attaining the goal $\mathcal{G}$ from any state $s \in \mathcal{S}$.

We will construct a collection of sets of states $\mathcal{S}_0, \ldots \mathcal{S}_q$, for some $q \leq |\mathcal{S}|$. The intuition behind these sets is that a state is in $\mathcal{S}_i$ if there exists a perfect-sensing strategy for attaining the goal in at most i steps, and if there is some possible instantiated transition graph for which i steps are actually required. We will not actually require this property in the current proof. However it provides the proper intuition, and it will reappear in the proofs of claims 3.5 and 3.12.

Define $\mathcal{S}_0$ to be the goal set $\mathcal{G}$. Clearly there is a perfect-sensing strategy that attains a goal state from any state in $\mathcal{S}_0$, requiring zero steps. Suppose that $\mathcal{S}_k$ has been defined, and that there exists a perfect-sensing strategy defined on the union $\bigcup_{i=0}^{k} \mathcal{S}_i$. The perfect-sensing strategy is assumed to attain a goal state from any state in the union without ever passing through any state in the complement $\mathcal{S} - \bigcup_{i=0}^{k} \mathcal{S}_i$. Define $\mathcal{S}_{k+1}$ to be the set of all states in this complement for which there exists some action that attains a state in $\bigcup_{i=0}^{k} \mathcal{S}_i$ in a single step. In other words,

$$\mathcal{S}_{k+1} = \left\{ s \in \mathcal{S} - \bigcup_{i=0}^{k} \mathcal{S}_i \;\middle|\; F_A(s) \subseteq \bigcup_{i=0}^{k} \mathcal{S}_i \text{ for some action } A = A(s) \right\}.$$

We need to show that $\mathcal{S}_{k+1}$ is not empty, unless $\mathcal{S} = \bigcup_{i=0}^{k} \mathcal{S}_i$. Once we establish this, then the existence of a perfect-sensing strategy on the union $\bigcup_{i=0}^{k+1} \mathcal{S}_i$ will be clear. In particular, this new perfect-sensing strategy is an extended version of the previous strategy. It executes the same actions as before for states in $\bigcup_{i=0}^{k} \mathcal{S}_i$, while executing the actions $A(s)$ for each $s \in \mathcal{S}_{k+1}$. Clearly this strategy attains a goal state from any state in the union $\bigcup_{i=0}^{k+1} \mathcal{S}_i$ without ever passing through states in the complement of this union.

Furthermore, since each set $\mathcal{S}_i$ is non-empty, there can be at most $|\mathcal{S}|$ of them.

Now let us show that $\mathcal{S}_{k+1}$ is indeed non-empty. Let us write $\mathcal{C}_k = \mathcal{S} - \bigcup_{i=0}^{k} \mathcal{S}_i$. Suppose that $\mathcal{S}_{k+1} = \emptyset$, but that $\mathcal{C}_k \neq \emptyset$. This says that for every state $s \in \mathcal{C}_k$ and every action A, the intersection of the forward projection $F_A(s)$ with $\mathcal{C}_k$ is non-empty. Said differently, for each state $s \in \mathcal{C}_k$, and each action A, there is an instantiation that causes s to traverse to a state in $\mathcal{C}_k$. This means that there is an instantiated transition graph for which the set $\mathcal{C}_k$ is completely disconnected from the goal. That violates the assumption of the claim, and thus we see that $\mathcal{S}_{k+1} \neq \emptyset$. ∎

The next claim establishes that a perfect-sensing strategy for attaining a goal need not be very long. The claim actually follows from the proof of the previous claim. However, for completeness we will prove it independently.

Claim 3.5 *Let $(\mathcal{S}, \mathcal{A}, \Xi, \mathcal{G})$ be a discrete planning problem, with Ξ being a perfect-sensing function. Suppose that there exists a guaranteed strategy for moving from some start state s to the goal set $\mathcal{G}$. Then this strategy requires no more than $r = |\mathcal{S}| - |\mathcal{G}|$ steps.*

Proof. This is a standard finite automaton argument. Suppose that more than r steps are required. Consider a possible trace of states that occur as the strategy is executed. This trace must then contain a subsequence of non-goal states in which the first and last state are the same state, say state $\hat{s}$. Let A_1 be the action executed when the system is first in state $\hat{s}$, and let action A_2 be the action executed when the system encounters state $\hat{s}$ at the end of this subsequence. Since sensing is perfect, the strategy will continue to be successful if action A_1 is replaced by action A_2. This change removes the subsequence from the trace, thus shortening this particular trace by at least one step. Repeatedly applying this procedure to all possible traces shows that the strategy need not require more than r steps. ∎

3.3 Perspective

We noted in section 3.2.3 that the general problem of planning optimal strategies on discrete spaces is very hard computationally. This suggests several different directions to take. One is to give up the notion of optimality. Another is to examine special cases, and to try to understand the characteristics that permit fast solutions. The next few sections will address these issues in the probabilistic setting.

Finally, as indicated earlier, for many problems the action transitions are not probabilistic but rather non-deterministic. In these situations, the Markov Decision model is not directly applicable. The approach for several years has been to compute what are often known as *guaranteed strategies*. These are strategies that are guaranteed to attain a goal state in a fixed number of steps, despite uncertainty. The strategies are computed by backchaining. In the perfect-sensing case, this amounts to using dynamic programming, with a value function that can only take on the boolean values 0 and 1. However, not all problems admit to guaranteed solutions. The latter sections of the chapter will look at how randomization may be used to solve some of these problems.

3.4 One-Dimensional Random Walk

In studying randomized strategies on discrete spaces, it is worthwhile to start by considering some very simple problems, such as the one-dimensional random walk. It turns out that the insight into convergence speeds that one gains from looking at a one-dimensional setting carries over to some extent into the general setting.

3.4.1 Two-State Task

The simplest possible non-trivial example is given by a system consisting of two states, with a probabilistic transition between these states. This was essentially the representation in the gear-meshing and parts-sieving examples earlier. For completeness, let us quickly review the results of the earlier discussion. Let us say that one of the states is the start state, and the other is the goal state, and that sensing is perfect. This means that whenever the system is in a state s, the sensor accurately reports that the system is in state s. If the probability of transiting from the start state to the goal state is p, then the expected time until the goal is attained is $1/p$. Indeed, this is a classic waiting time problem: the probability that the goal is attained on the k^{th} try is $p\,q^{k-1}$, where $q = 1-p$. In particular, for fixed p, convergence is exponentially fast in the number of tries. [This is also known as linear convergence or geometric convergence, since the ratio of successive error terms is bounded by a constant less than one.]

A slightly more complicated problem is given if the sensing function is not perfect. Different variations are possible. One possibility is that sometimes the sensor will correctly register the state of the system, while at other times the sensor cannot

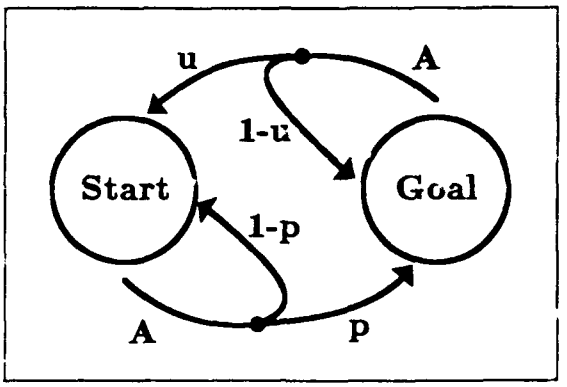


Figure 3.9: A two state Markov chain with probabilistic transitions. The chain represents an approximation to the gear-meshing and sieving examples of section 1.2, given that the action to be executed is fixed.

distinguish between the two states. If p_{sense} is the probability of recognizing that the system is in the goal, then the probability of entering and recognizing entry into the goal is $p\,p_{sense}$, assuming independence between the actions and the sensors. This raises the expected execution time by at most a factor of $1/p_{sense}$. Another possibility is that the sensing function can never distinguish between the start and the goal when the system is at the goal. In this case one cannot guarantee that the goal will be attained, but one may be able to say something about the probability of attaining the goal in a specific number of steps.

At issue is what happens to the system if it is in the goal and one executes the action designed to move to the goal. Specifically, one is interested whether the system stays in the goal or whether it can jump back out of the goal. In the gear-meshing example the question amounts to deciding whether spinning the gears when they are meshed can cause them to disengage. In the sieving example, the question amounts to deciding whether shaking the system after the object has fallen through the sieve can cause it to jump back up above the sieve. See figure 3.9 for a probabilistic description. The probability of moving out of the goal is given by u. This is zero if the system remains forever in the goal under the action A, and non-zero otherwise.

The ideal situation is that u is zero, in other words, that the goal is not ever exited once attained. In this case, as we mentioned above, the probability of not attaining the goal in k attempts is q^k, where $q = 1 - p$ is the probability that the system stays in the start state when action A is executed. So, if one wants the probability of failure to be less than some constant ϵ, then one should choose k to be bigger than $\log \epsilon / \log q$.

The worst case occurs when u is one, that is, when the goal is immediately exited after having been entered. Define p_k to be the probability that the system is in the goal on the k^{th} try, and q_k to be the probability that the system is not in the goal.

Then the following system of equations holds:

$$\begin{aligned} q_{k+1} &= q\,q_k + p_k, \\ p_{k+1} &= p\,q_k, \end{aligned}$$

with boundary conditions

$$q_0 = 1, \qquad p_0 = 0.$$

Since $q_k + p_k = 1$, we see that the strategy which attains the goal with highest probability consists of a single attempt. In order to see this, notice that $p_0 = 0$ and that $p_1 = p$. Observe further that $p_k > 0$ for all $k > 0$. Thus $q_k < 1$ for all $k > 0$. This in turn says that $p_k < p$ for all $k > 1$. In other words, after the first trial the probability of success decreases.

If one does repeatedly try to attain the goal, so that k becomes very large, then q_k and p_k approach a limiting distribution, namely

$$\begin{aligned} q_k &= \tfrac{1}{1+p}, \\ p_k &= \tfrac{p}{1+p}. \end{aligned}$$

Let us briefly consider the general case. All this material is standard in the theory of Markov chains. See, for instance, [FellerI] and [KT1] for further introductions. Denote the start state as state 1 and the goal as state 2. Let $\mathbf{P} = (p_{ij})$ be the probability transition matrix, where p_{ij} is the probability of transiting from state i to state j in a single step. We have that

$$\mathbf{P} = \begin{pmatrix} 1-p & p \\ u & 1-u \end{pmatrix}.$$

The k^{th} power of this matrix, $\mathbf{P}^k$ describes the k-step transition probabilities. If the row vector $\boldsymbol{\pi}_0$ describes the initial probability distribution over the system states, then $\boldsymbol{\pi}_k = \boldsymbol{\pi}_0 \mathbf{P}^k$ describes the resulting probability distribution after k steps. In our case $\boldsymbol{\pi}_0 = (1,0)$, meaning that the system starts off in state 1. The theory of Markov chains tells us that as the number of steps gets large $\boldsymbol{\pi}_k$ approaches a limiting distribution $\boldsymbol{\pi}$, which is a left eigenvector of the matrix $\mathbf{P}$, with eigenvalue 1. Furthermore, under fairly simple conditions (such as non-periodicity), the chain converges to this distribution at the rate λ^k, where λ is the largest eigenvalue whose norm is less than one (all eigenvalues have norm no more than one). So convergence is exponentially fast in the number of steps taken. It is clear that the vector $\boldsymbol{\pi} = (u/(u+p), p/(u+p))$ is a left eigenvector of the matrix $\mathbf{P}$, with eigenvalue 1. Thus $\boldsymbol{\pi}$ forms the limiting distribution as one repeatedly executes action A. Furthermore, the eigenvalue other than 1 is $\lambda = 1 - p - u$, and convergence occurs geometrically fast, with λ as base. Indeed, if we write the difference at any point in time between the limiting distribution and the current distribution as $\mathbf{e}_k = \boldsymbol{\pi} - \boldsymbol{\pi}_k$, then $\mathbf{e}_k$ is of the

form $(\epsilon, -\epsilon)$, for some ϵ between -1 and 1. Furthermore, $\mathbf{e}_{k+1} = \mathbf{e}_k \mathbf{P}$, by definition of the limiting distribution. Performing this multiplication, we see that $\mathbf{e}_{k+1} = \lambda \mathbf{e}_k$, as one would hope. This also shows that the strategy which maximizes the probability of attaining the goal, given that one starts out in the start state, is given by a single application of action A. Further applications of A only reduce the probability of being in the goal, from p eventually down to the stable distribution value of $p/(u+p)$. Of course, if one isn't sure whether the system initially starts in the (so-called) start state or in the goal state, then a single application of A may not be the right thing.

If we apply this analysis to the case that u is zero, we see that the system has eigenvalues 1 and q, and a limiting distribution of $\boldsymbol{\pi} = (0,1)$. This says that the goal is eventually attained, and that convergence is geometric with base q, agreeing with our earlier calculations. In the case that u is one, the eigenvalues are 1 and $-p$. Again, the limiting distribution is $\boldsymbol{\pi} = (1/(1+p), p/(1+p))$, as we saw earlier, and convergence to this distribution is geometric with base $-p$. The negative sign indicates oscillatory behavior of the error vector.

For a given action we now have a means of computing the probability of winding up in the goal on any given step. Or, more generally, without knowing anything about the initial distribution that determines the state of the system, we can say that after sufficiently many applications of action A the system will attain a stable distribution. In particular, after sufficiently many steps the goal will be attained with probability close to $p/(u+p)$. While this is a far cry from guaranteeing that the goal will be attained, it is considerably better than claiming that the task is not doable in the absence of a guaranteed strategy. In particular, if the goal represents the preconditions to some other task, then one has a means of at least probabilistically meeting those preconditions, and of passing on a probability of their having been met to the next task. Said differently, one can think of the repeated execution of action A as randomizing between two states, of which only one permits solution of some additional task. With good sensing the randomization is not needed, but with no sensing, the randomization offers a means of solving the task without knowing whether the system first starts off in the goal or in the (so-called) start state.

Suppose that several different actions are possible. Then this analysis provides a means for comparing the actions in terms of their probabilities of success or in terms of their convergence times.

Furthermore, the approach just outlined applies to a general Markov chain. The size of the matrices changes, but the comments regarding limiting distributions and convergence times continue to hold. We can thus imagine analyzing and comparing different strategies for solving a sensorless task formulated as a probabilistic problem on a discrete state space.

3.4.2 Random Walks

In order to motivate the analysis of random walks, consider the task of moving a peg into a hole. Suppose that we are interested in generating a simple feedback loop, that senses the position of the peg relative to the hole, then moves the peg to decrease the

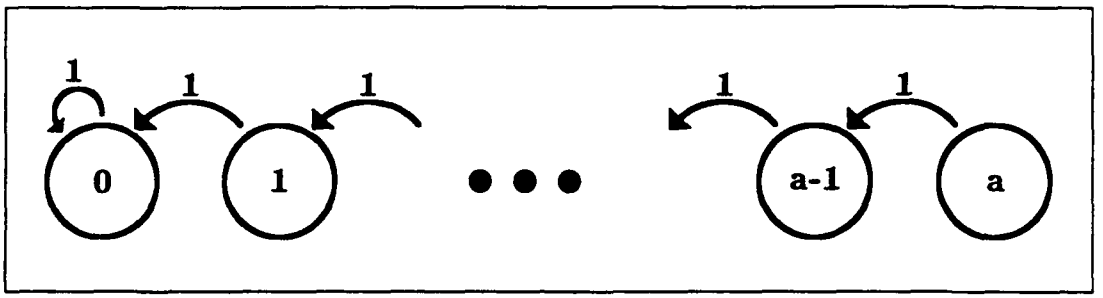


Figure 3.10: This is a deterministic random walk. The system moves towards the left one state during each time step.

distance from the hole. In the case of perfect control and perfect sensing, the peg will always move towards the hole. However, if sensing and control are subject to error, then the sensors may occasionally suggest the wrong direction in which to move, and the motions executed may occasionally move in the wrong direction or perhaps accidentally slide over the hole. Recall the physical peg-in-hole example of section 1.1 and the analysis of section 2.4. In other words, the motion at any point is not guaranteed to move towards the hole, but has some chance of moving in a different direction. This sets the stage naturally for processes that may be approximated by random walks. These are in general multi-dimensional, but often it is enough to consider some one-dimensional quantity, such as the distance of the peg from the hole. A more direct example is given by a two-dimensional peg-in-hole problem, in which the peg is moving on a one-dimensional edge near the mouth of the hole.

Another motivation for studying random walks is given by the sensorless tasks discussed in section 1.4. Here the question may be one of choosing a sequence of probabilistic actions that should attain some desired goal. The choice may be deterministic, so that the random character of the system arises solely from the probabilistic actions, or the choice of actions may itself involve random decisions at execution time.

In summary, random walks on graphs arise naturally due to uncertain sensing, uncertain control, and purposeful randomization. We are interested in this section in determining convergence properties of one-dimensional random walks. An understanding of these properties will aid in constructing strategies for more general tasks.

Figure 3.10 shows a simple one-dimensional random walk. The state space consists of $a+1$ states, labelled $0, 1, \ldots, a$. The arrows emanating from each state indicate the possible transitions out of that state at any given step of the process. The arrows are labelled with the probability of their occurrence. State 0 is the goal. This is actually a deterministic random walk: At each step, if the process is in state k, then it will transit to state $k-1$ with probability one. Once the process has entered state 0, it remains there. In short, for this deterministic random walk the expected time to

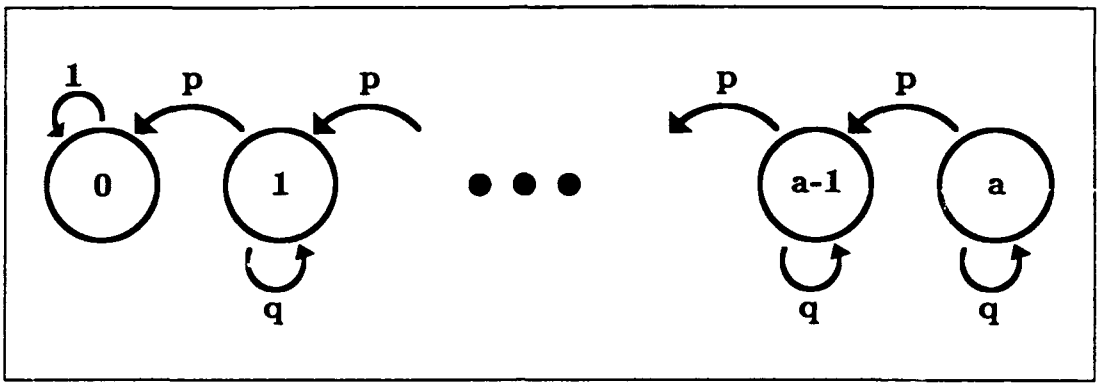


Figure 3.11: This is a random walk, in which the system moves left with probability p, and sits still with probability $q = 1 - p$.

reach the goal from state k is just k, the distance to the origin. This is the type of behavior one expects with perfect control or sensing.

A slight variation is given by the random walk in figure 3.11. In this example the transition from state k to state $k - 1$ only has probability p, while with probability $q = 1 - p$ the process remains in state k. An example of such a process might be a series of sieves stacked one above the other (recall section 1.2). Once the object has passed through one sieve, it will not move back up, but it need not immediately pass through the next sieve. Another example mentioned earlier was the task of closing a desk drawer that is slightly wedged. In many cases it may be enough to keep trying to push the drawer shut, without ever having to pull it out. The probability p models the probability of selecting a pushing force that actually closes the drawer further.

The expected convergence time for the process is now k/p, if it initially starts in state k. One sees therefore that the transition probability acts almost like a velocity. In this example the velocity is p; in the previous example it was 1. Later we will generalize this notion of velocity to a more encompassing setting.

One final comment concerns the search for paths to the goal. If one simply employed a connectivity analysis, one would see that the goal is reachable from state k by a sequence of length k. The probability that this precise sequence will actually be executed is p^k, which suggests horrible convergence times. Fortunately, however, because progress along the chain cannot be arbitrarily undone, the actual convergence times are much faster.

We will now derive the convergence times of a fairly general random-walk. For the most part, we will follow [FellerI] in this analysis (see in particular pp. 348-349), although our boundary conditions are slightly different. As usual, transitions are possible only to neighbor states, and we will assume that the probabilities are the same for all interior states. In particular, p is the probability of moving left, and $q = 1 - p$ is the probability of moving right. We are not considering self-transition

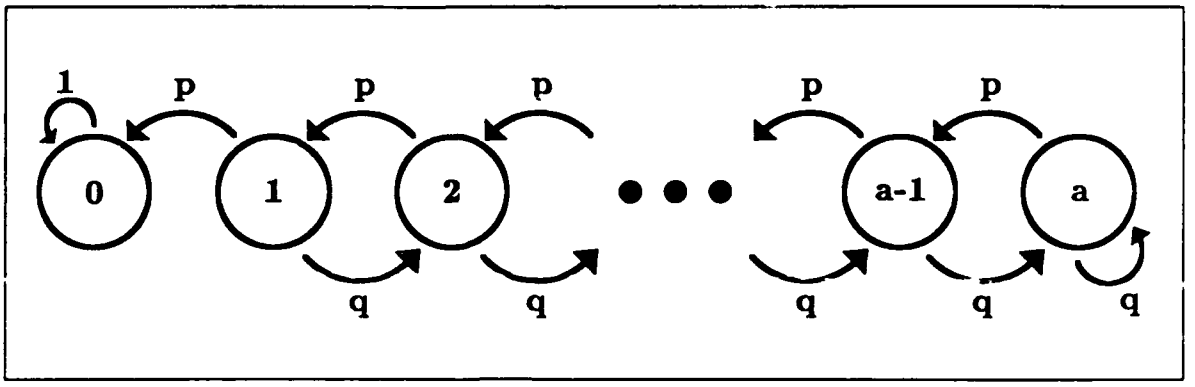


Figure 3.12: This is a random walk, in which the system moves left with probability p, and right with probability $q = 1 - p$. The random walk stops in state 0 and reflects at state a.

probabilities. If these are included then the results are nearly identical. In the case that $p = q$ the expected durations are scaled by $1/(p+q)$ from those given here. In the asymmetric case, the results are identical to those given here, except that q and p no longer add to one. We assume further that the process stops in state 0, and reflects at state a. In other words, instead of moving right with probability q from state a, the process simply stays in state a with probability q. See figure 3.12.

Now let D_k be the expected time to reach the goal (state 0), given that the system starts in state k, with $0 \le k \le a$. Suppose the system starts in state k with $k < a$, and consider the results of its first step. With probability p the system will move to the left, at which point the remaining expected time is D_{k-1}, and with probability q the system moves to the right, whereupon the remaining expected time to reach the goal is D_{k+1}. This establishes the following difference equation.[2]

$$D_k = q\,D_{k+1} + p\,D_{k-1} + 1, \qquad 0 < k < a, \tag{3.5}$$

with boundary conditions

$$D_0 = 0, \quad D_a = q\,D_a + p\,D_{a-1} + 1. \tag{3.6}$$

Let us first suppose that $p \neq q$. Then a general solution to equation (3.5) is given by

$$D_k = \frac{k}{p-q} + A + B\left(\frac{p}{q}\right)^k, \qquad \text{when } p \neq q, \tag{3.7}$$

where A and B are arbitrary constants. In our case, these are determined by the boundary conditions (3.6). In particular,

[2]These equations follow Feller, but with different boundary conditions.

$$D_0 = 0 \Rightarrow A + B = 0,$$

and

$$\begin{aligned} D_a &= q D_a + p D_{a-1} + 1 \\ \Rightarrow \quad D_a &= D_{a-1} + \tfrac{1}{p}, \end{aligned}$$

which says that

$$\frac{a}{p-q} + A + B\left(\frac{p}{q}\right)^a = \frac{a-1}{p-q} + A + B\left(\frac{p}{q}\right)^{a-1} + \frac{1}{p}.$$

It follows that

$$B = -\frac{q}{(p-q)^2}\left(\frac{q}{p}\right)^a,$$

from which we see that the solution to (3.5) and (3.6) is given by

$$D_k = \frac{k}{p-q} + \frac{q}{(p-q)^2}\left(\frac{q}{p}\right)^a - \frac{q}{(p-q)^2}\left(\frac{q}{p}\right)^{a-k}. \tag{3.8}$$

It is useful to rewrite this solution as

$$D_k = \frac{k}{p-q} + \frac{q}{(p-q)^2}\left(\frac{q}{p}\right)^a \left[1 - \left(\frac{p}{q}\right)^k\right]. \tag{3.9}$$

Suppose now that $p > q$ (so, in some sense, the "natural drift" is towards the origin). Then the factor $1 - (p/q)^k$ is negative. So

$$D_k \leq \frac{k}{p-q}, \qquad 0 \leq k \leq a,$$

and we see that convergence is essentially linear in the distance from the origin. In fact, if a is large and $k \ll a$, then $D_k \approx k/(p-q)$.

Now, suppose that $q > p$, so the "natural drift" is away from the goal. This time the factor $1 - (p/q)^k$ is positive, and the factor $(q/p)^a$ becomes significant. Indeed, for large a (and moderate to large k), the expected durations are essentially

$$D_k \approx \frac{q}{(p-q)^2}\left(\frac{q}{p}\right)^a.$$

In other words, convergence is exponential in the length of the random walk. For small k, this time is reduced slightly, but it is still of the same order.

Finally, let us consider the case for which $p = q = 1/2$. Then the general solution to the difference equation (3.5) becomes

$$D_k = -k^2 + A + B\,k.$$

The first boundary condition implies that A is zero, while the second boundary condition says that $D_a = D_{a-1} + 2$, from which we see that $B = 2\,a + 1$. So, the complete solution is

$$D_k = k\,(2\,a + 1 - k).$$

In other words, the convergence times are essentially quadratic. In particular, for values of k comparable to the length of the chain a, the convergence times are essentially a^2, whereas for smaller values of k the convergence times are on the order of $k\,a$.

These observations establish the following

Claim 3.6 *Consider a random walk on the state space $0, 1, \ldots, a$, with reflection at a. Let p be the probability of moving left one unit, and let $q = 1 - p$ be the probability of moving right one unit. Then the maximum expected time to attain the origin is linear in a, quadratic in a, or exponential in a, depending on whether $p > q$, $p = q$, or $p < q$, respectively.*

Furthermore, for a fixed starting location k, the expected time to attain the origin from k approaches $k/(p-q)$ as $a \to \infty$ if $p > q$, and approaches infinity if $p \leq q$.

We see then that it is important for a random walk to drift in the correct direction. If at each point in time the tendency is to move towards the goal, then the random walk behaves very much like a deterministic process. Specifically, the expected time to reach a goal is essentially the distance to the goal divided by the expected velocity at which the process is moving. In the random walk case, the quantity $p-q$ measures this expected velocity. On the other hand, if the expected velocity is pointing in the wrong direction, then the goal will still be attained eventually (assuming that the state space is finite), due purely to randomness. Now, however, the exponential character of having to perform several operations, each of which succeeds only with some probability, becomes dominant.

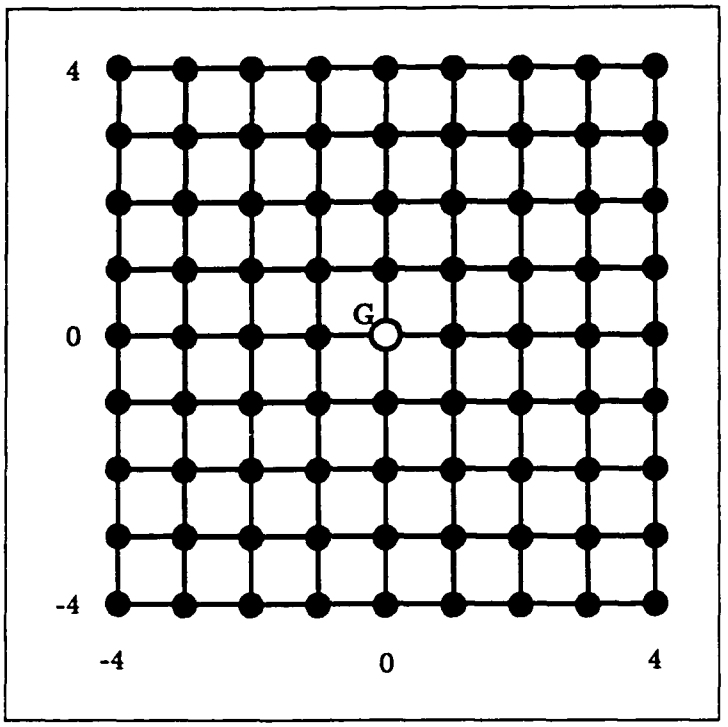


Figure 3.13: A bounded two-dimensional grid, with the goal at the origin.

3.4.3 General Random Walks

Thus far we have looked only at random walks for which the transition probabilities are identical over all the (interior) states. If one varies the probabilities, then one can obtain mixtures of the three types of random walks discussed thus far. For instance, if some of the local velocities point away from the origin, whereas most point towards the origin or at least are zero, then one can obtain convergence times that are worse than linear or quadratic, but do not yet approach the exponential character of a random walk for which all velocities point away from the origin. Examples in which this type of behavior arises naturally are given by random walks in higher dimensions. For instance, consider the two-dimensional grid of figure 3.13. Consider a two-dimensional random walk on this grid, in which transitions occur only to immediate neighbor points, each with probability 1/4, and reflection occurs at the boundary. Suppose the origin is the goal, and consider the one-dimensional quantity given by distance from the origin, measured as Manhattan distance. For points off the horizontal and vertical axes, two of the four possible transitions decrease the distance to the goal, while two increase the distance from the goal. The expected change in distance from the origin is in fact zero, that is, the "drift velocity" relative to the origin is zero. On the other hand, for points on either of the axes, only one transition decreases the distance from the origin, while three increase the distance.

The expected change in distance is in fact $+1/2$, that is, the drift velocity points away from the origin. Fortunately a point on one of the axes has probability $3/4$ of either moving off the axis or of moving closer to the goal. Thus, even though the natural drift on the axes is away from the origin, the system cannot get stuck on the axes, so that one does not see an exponential convergence time. This particular mixture of velocities that are either zero or point away from the origin yields a maximum expected convergence time that is on the order of $a^2 \log a$. Here the grid has edge length a (see [Montroll]). This is slightly worse than the quadratic convergence time for the one-dimensional random walk in which all the (interior) local velocities were zero, but not so bad as the case in which all the velocities actually pointed away from the goal. In higher dimensions, the mixture gets slightly worse, so that on a grid in d dimensions the maximum expected convergence time is on the order of a^d, which is the grid size. All of these times are still polynomial in a.

3.4.4 Moral: Move Towards the Goal on Average

In the previous examples the natural drift was either zero or it pointed away from the origin. In order to attain expected velocities that point towards the origin, one needs some mechanism that naturally skews the random walk towards the goal. If we think of the random walk as arising from some underlying mechanical task, then this direction must be given by either the mechanics of the task or by the use of sensors. For instance, the goal might physically be located at the bottom of some trough or funnel. Alternatively, if the sensors provide enough useful information then one may be able to guide the system towards the goal on average.

The moral is that in order to obtain reasonable convergence times for some task, one should try locally to make progress on the average. In fact, one need not guarantee progress at every location or at every moment. However, if there are a reasonable number of locations for which progress occurs on the average, then convergence will be reasonably quick. This view of the world is considerably different from the one that insists on guarantees at every step.

Given these observations, the study of robotics, in particular the study of automating the solution to assembly tasks, becomes one of finding a proper mixture of sensing, motion, and randomization, that ensures progress on the average. Other issues include the definition of progress itself, plus numerous details that delineate the scope of the approach. The remainder of the thesis will address some of those issues.

3.5 Expected Progress

The first issue that needs to be addressed is the definition of expected velocity in the setting of a general Markov chain. For the one-dimensional random walk, with transitions only to neighbors, this was fairly straightforward, but we need a precise definition for the general case. The second issue is whether these so-called velocities

behave nicely, in particular, whether increasing the expected velocity towards the goal at some point, reduces the expected time to reach the goal. This is certainly true in the deterministic case, and one would like it to hold as well for the probabilistic case.

The basic motivation for defining a velocity is to be able to discuss the speed with which progress towards the goal is made. In turn, this allows us to analyze and compare different randomized strategies. So let us suppose that we have a finite state space with states $s_0, s_1, \ldots, s_n$, with a single goal state s_0. Let us assume that we are given a labelling of these states, that is, to each state s_i there is associated a one-dimensional number ℓ_i (perhaps a real number). The idea is to view these labels as defining a progress measure, then to define expected velocities in terms of the expected progress determined by this progress measure.

To make this precise, let $\mathbf{P} = (p_{ij})$ be the probability transition matrix for some chosen strategy for moving from non-goal states to the goal. We are assuming that the task is formulated in such a way that the effect of our strategy may indeed be described probabilistically at each step of execution. Then the average or expected velocity at state s_i is defined to be

$$v_i \stackrel{\text{def}}{=} \sum_{j=1}^{n} p_{ij}\,(\ell_j - \ell_i). \tag{3.10}$$

The sum on the right just measures the average displacement from state s_i, measured in terms of the labelling, caused by a single step of the strategy. Thinking of each step as being one unit of time then yields the average velocity. Note that we can rewrite equation (3.10) as

$$v_i = \sum_{j=1}^{n} p_{ij}\,\ell_j \;-\; \ell_i. \tag{3.11}$$

That is the definition, and here is the main claim of this section. It establishes the usefulness of the definition of expected velocity.

Claim 3.7 *Consider a Markov chain with states $\{s_i\}$ and probability transition matrix (p_{ij}). One of the states, say s_0, is a goal state. By this we mean that all states eventually transit to s_0. Suppose further that $\{\ell_i\}$ is a labelling of the states which is zero at the goal state and positive elsewhere. Let $\ell = \max_i\{\ell_i\}$ be the maximum label, and let $v = \max_i\{v_i\}$ be the maximum expected velocity defined by this labelling. Finally, let $D = \max_i\{D_i\}$ be the maximum expected time to reach the goal, where D_i is the expected time to reach the goal given that the system starts in state s_i.*
The claim is that whenever v is negative, then

$$D \leq -\frac{\ell}{v}.$$

Said differently, the maximum expected time to reach the goal is bounded by the maximum distance to the goal (measured by the labelling), divided by the minimum expected velocity of approach to the goal:

$$\max_i D_i \leq \frac{\max_i\{\ell_i\}}{\min_i\{-v_i\}}.$$

In fact, for each state, the expected time to reach the goal is bounded by the state's label divided by the minimum expected approach velocity:

$$D_i \leq \frac{\ell_i}{\min_i\{-v_i\}}.$$

Proof Strategy. The basic strategy of the proof is to first establish that if the expected velocity is the same at each state, then the expected time to attain the goal is just the state label divided by this expected velocity. We then show that any Markov chain satisfying the hypotheses of the claim may be formally modified so that the expected velocity is the same at each state. [This modification is purely a proof technique and has nothing to do with the underlying physical process.] Finally, we show that the modified Markov chain may be transformed back into the original chain in such a way that the expected convergence times decrease or remain the same. This will establish the claim.

Proving the claim will require a little bit of work, but it is intuitively desirable and clear. The claim shows that under suitable conditions a general Markov chain behaves very much as does the one-dimensional random walk discussed in section 3.4.2. Specifically, if a randomized strategy can ensure that on the average it decreases sufficiently quickly some measure of distance from the goal, then the expected time to attain the goal will be linear in that measure. From a planning point of view this suggests two problems: finding strategies that make local progress relative to a given progress measure, and finding useful progress measures.

In order to establish the claim, we will state and prove several other simple propositions. These will provide further intuition regarding the nature of progress measures within randomized strategies. First, let us turn the problem around. Instead of starting with a labelling of the state space and determining a strategy for making progress relative to the labelling, suppose one started with a randomized strategy. In particular, suppose a randomized strategy is given that turns the state space into a Markov chain that eventually converges to some goal state. It is natural to ask whether there is a labelling of the state space relative to which the strategy may be perceived as making progress. The answer is of course yes. If one simply labels the states with their expected times until success, then the induced expected velocities will all be -1. Essentially the labelling spreads out the states far enough that the distance between them corresponds precisely to the difference in expected times to reach the goal. Of course, the labels may now be very large numbers! We prove this observation in the following claim.

Claim 3.8 *Given a Markov chain $(\{s_i\},(p_{ij}))$ for which all states eventually transit to some goal state s_0, label each state s_i with D_i, the expected time to attain the goal given that the system starts in state s_i. Relative to this labelling the induced expected velocities $\{v_i\}$ are all -1 (for non-goal states).*

Proof. We have that $D_i = \sum_{j=0}^{n} p_{ij} D_j + 1$. This is just a generalization of the argument used to establish the convergence times for random walks (section 3.4.2). Rewriting this, we see that $\sum_{j=0}^{n} p_{ij} D_j - D_i = -1$. Interpreting the expected times as labels, we see by (3.11) that the left-hand side of this equation is just v_i, which establishes the claim. ∎

This says that labellings are a natural means of characterizing a strategy's behavior. It also indicates that the search for a useful labelling is futile, since any strategy can be made to appear to converge quickly relative to a suitable labelling. It is in fact more appropriate to view the situation in reverse. If one is interested in convergence speeds of a particular type, then one should look at labellings whose labels do not exceed the desired convergence times. For any such labelling one can then determine whether a strategy exists that makes rapid progress. Indeed, in many cases a natural labelling may be apparent, such as one given by the distance or distance squared from some goal.

Finding a strategy given a labelling essentially entails choosing the $(n+1)^2$ probabilities $\{p_{ij}\}$, subject to the constraints $v_i < 0$, and $\sum_{j=0}^{n} p_{ij} = 1$, for all $i = 1, \ldots, n$. If choosing these probabilities can be done independently for each state s_i, then the existence of a fast strategy relative to a labelling may be ascertained very quickly, since all the computations and constraints are local to each state s_i. In many cases, however, the strategy cannot be determined locally. For instance, the action performed in a given state will depend on a sensor value returned when the system is in that state. Since different states can give rise to the same sensor value, a strategy based on sensed values will necessarily couple the p_{ij} at different states. We will see the significance of this topic later, both in this chapter and in chapter 5. Indeed it will turn out that for simple labellings, such as distance from the goal, average progress cannot always be guaranteed for every state in the system. Instead, one naturally gets mixtures of states, some for which rapid progress is possible and some for which it is not, just as we did for the two-dimensional random walk discussed in section 3.4.3.

An immediate corollary to Claim 3.8 is the following.

Corollary 3.9 *If relative to some labelling $\{\ell_i\}$, all the expected velocities are equal to a negative constant v_{const}, then the expected times to reach the goal are given by $D_i = -\ell_i / v_{const}$.*

Proof. Using the expression (3.11), we have that at each state s_i

$$\begin{aligned} v_{const} &= v_i \\ &= \sum_{j=1}^{n} p_{ij} \ell_j - \ell_i. \end{aligned}$$

Relative to a new labelling $\{\ell'_i\}$ given by $\ell'_i = \ell_i / (-v_{const})$, one observes that:

$$-1 = \sum_{j=1}^{n} p_{ij}\,\ell'_j \;-\; \ell'_i.$$

By the proof of claim 3.8, it must therefore be the case that $\ell'_i = D_i$ for all states s_i. (Uniqueness of the D_i follows from the assumption that all the states eventually transit to the goal. See also chapter 10 of [KT2].) This establishes the corollary. ∎

This corollary is useful in conjunction with the next lemma, which establishes that we can always modify a finite Markov chain whose expected velocities are negative so that its expected velocities are all equal to some non-zero negative constant. In particular, we will show that if the average velocity at some state is negative then that state's average velocity may be increased (that is, its absolute value may be decreased) by changing into self-transitions some of the transitions that point to states with lower labels. [Note that we are not claiming anything about whether the underlying physical process may be changed.] For a finite Markov chain with negative expected velocities this immediately implies that the chain may be modified so that all expected velocities are some negative constant. As we outlined on page 130, this is useful as a proof device for the proof of claim 3.7.

Lemma 3.10 *Consider a labelled Markov chain* $(\{s_i\},(p_{ij}),\{\ell_i\})$. *Suppose that the expected velocity* v_k *at some state* s_k *is negative. Let* α *satisfy* $v_k \le \alpha \le 0$. *Then one can modify the* k^{th} *row of the probability transition matrix* (p_{ij}) *so that the velocity at* s_k *becomes* α. *Furthermore, one need only increase* p_{kk} *and commensurately decrease* p_{kj} *for values of* j *for which* $\ell_j < \ell_k$.

Proof. Let $\Delta v = v_k - \alpha$. Then $v_k \le \Delta v \le 0$.

Since v_k is negative, we have that $\ell_j < \ell_k$ for at least one j (see the definition, equation (3.10)). Furthermore, taking all these s_j together, we must have that

$$\sum_{\ell_j < \ell_k} p_{kj}\,(\ell_j - \ell_k) \le v_k \le \Delta v.$$

For the purposes of argument it is enough to assume that there is one $j = j_0$ for which $p_{kj_0}\,(\ell_{j_0} - \ell_k) \le \Delta v$. The general case follows readily from this.

Now define a new probability transition matrix (p'_{ij}) which is identical to (p_{ij}), except for p'_{kk} and p'_{kj_0}. Specifically, let $p'_{kk} = p_{kk} + p$ and $p'_{kj_0} = p_{kj_0} - p$, where $p = \Delta v/(\ell_{j_0} - \ell_k)$. One verifies that $0 \le p \le p_{kj_0}$, so the construction makes sense. It is easily seen that the induced velocity v'_k equals α, thus establishing the lemma. ∎

As an aside, one notes that the lemma holds with proper modifications for positive expected velocities, although this is less useful in the current context.

Now we need a lemma that goes in the other direction. Specifically, if we increase the average velocity with which the goal is approached at some point, then we would like to know that the expected time to reach the goal decreases. From our random walk example, and given the phrasing of this claim, this is intuitively clear, but in

a general setting some proof is required. The following lemma forms the core of our proof of Claim 3.7.

Lemma 3.11 *Consider a Markov chain with $n+1$ states $s_0, s_1, \ldots, s_n$ and probability transition matrix (p_{ij}). Suppose state s_0 is the goal state (this means that all states eventually transit to s_0 and remain there). Let D_i be the expected time to reach s_0 given that the system starts in state s_i. Now consider two states s_x and s_y for which $D_x > D_y$. Construct a new Markov chain on the same state space with a modified probability transition matrix (p'_{ij}) that is almost identical to (p_{ij}). It differs in that $p'_{xx} = p_{xx} - p$ and $p'_{xy} = p_{xy} + p$, where p is any number satisfying $0 \leq p \leq p_{xx}$. If $\{D'_i\}$ are the new expected times to reach the goal, then $D'_i \leq D_i$, for all states.*

Furthermore, if p is non-zero, then $D'_x < D_x$.

Proof. The proof is long, although the idea is simple: Separate the behavior of the system into two parts, namely what happens at all states but state s_x, and what happens at state s_x. The behavior of the new system changes only at s_x (although the expected convergence times may change throughout the system), and intuitively that change only increases the probability of moving closer to the goal. Thus the expected convergence times should decrease. All this makes sense if we think of expected convergence times as labellings akin to distance measures.

And now for the details.

Let g_i be the probability that starting in state s_i the system reaches state s_0 before it reaches state s_x. This probability is well-defined for all states. Also, note that $g_0 = 1$ and $g_x = 0$.

Let D_i^x be the expected time to reach state s_x from state s_i, given that the system reaches s_x before s_0.

Let D_i^0 be the expected time to reach state s_0 from state s_i, given that the system does not pass through s_x.

And, let $D_i^{x,0}$ be the expected time to reach either state s_x or state s_0 from state s_i before reaching the other.

One observes that $D_i^{x,0} = g_i\, D_i^0 + (1 - g_i)\, D_i^x$, and that $D_i = g_i\, D_i^0 + (1 - g_i)\,[D_i^x + D_x]$.

Then for each non-goal state s_i, we have that

$$D_i = 1 + \sum_{j=0}^{n} p_{ij}\, D_j \tag{3.12}$$

$$= 1 + \sum_{j=0}^{n} p_{ij}\, \left[g_j\, D_j^0 + (1 - g_j)\,[D_j^x + D_x]\right] \tag{3.13}$$

$$= 1 + \sum_{j=0}^{n} p_{ij}\, D_j^{x,0} + \sum_{j=0}^{n} p_{ij}\,(1 - g_j)\, D_x. \tag{3.14}$$

Now, if we makes changes to p_{xx} and p_{xy} as suggested, then the expected durations $\{D_i\}$ will change, but all of the quantities $\{g_i\}$ and $\{D_i^{x,0}\}$ will remain the same. To

see this, observe that when $i \neq x$, g_i depends only on transitions at states other than state s_x. None of these transitions are affected by the changes to p_{xx} and p_{xy}. A similar argument applies to $\{D_i^{x,0}\}$ for $i \neq x$. Finally, observe that $g_x = 0$ always, and thus that $D_x^{x,0} = D_x^x = 0$ always.

Let us write out equation (3.14) for the state s_x, and simplify to get an expression for D_x:

$$D_x = 1 + \sum_{j=0}^{n} p_{xj} D_j^{x,0} + \sum_{j=0}^{n} p_{xj} (1 - g_j) D_x.$$

So, solving for D_x,

$$D_x \left[1 - \sum_{j=0}^{n} p_{xj} (1 - g_j) \right] = 1 + \sum_{j=0}^{n} p_{xj} D_j^{x,0},$$

and thus:

$$D_x = \frac{1 + \sum_{j=0}^{n} p_{xj} D_j^{x,0}}{1 - \sum_{j=0}^{n} p_{xj} (1 - g_j)}. \tag{3.15}$$

Now, let us introduce an artifice, by defining $\{D_i'\}$ and $\{p_{ij}'\}$ to be functions of p, where $0 \leq p \leq p_{xx}$. As mentioned already, these are the only quantities that change. In particular, we have that

$$p_{ij}'(p) = \begin{cases} p_{xx} - p, & \text{if } i = j = x \\ p_{xy} + p, & \text{if } i = x \text{ and } j = y \\ p_{ij}, & \text{otherwise.} \end{cases}$$

Substituting these changes into equations (3.14) and (3.15), and noting that $D_x^{x,0} = 0$, we have that

$$D_i'(p) = 1 + \sum_{j=0}^{n} p_{ij} D_j^{x,0} + \sum_{j=0}^{n} p_{ij} (1 - g_j) D_x'(p), \qquad \text{if } i \neq x, i \neq 0. \tag{3.16}$$

$$\begin{aligned} D_x'(p) &= \frac{1}{f(p)} \left\{ 1 + \sum_{\substack{j=0 \\ j \neq y \\ j \neq x}}^{n} p_{xj} D_j^{x,0} + (p_{xy} + p) D_y^{x,0} + (p_{xx} - p) D_x^{x,0} \right\} \\ &= \frac{1}{f(p)} \left\{ 1 + \sum_{j=0}^{n} p_{xj} D_j^{x,0} + p D_y^{x,0} \right\}, \end{aligned} \tag{3.17}$$

where (recalling that $g_x = 0$)

$$
\begin{aligned}
f(p) &= 1 - \sum_{j=0}^{n} p'_{xj}(p)\,(1 - g_j) \\
&= 1 - \sum_{\substack{j=0 \\ j \neq y \\ j \neq x}}^{n} p_{xj}\,(1 - g_j) \;-\; (p_{xy} + p)\,(1 - g_y) - (p_{xx} - p) \\
&= \sum_{j=0}^{n} p_{xj}\, g_j \;+\; p\, g_y. \qquad (3.18)
\end{aligned}
$$

Notice that $f(p)$ is always positive, in particular, it is never zero, so these equations make sense. To see that $f(p)$ cannot be zero, first observe that $f(p) \geq f(0) \geq 0$. Then observe that $f(0)$ is just the probability that if the system starts in state s_x it will reach the goal s_0 before reencountering state s_x. If this quantity were indeed zero, then the goal would be unreachable from state s_x, violating our connectivity assumption (see also section 3.2.7).

In order to establish the lemma one needs to verify that $D'_i(p) \leq D'_i(0)$ for all p in the range $0 \leq p \leq p_{xx}$. From equation (3.16), it is clear that whenever $D'_x(p) \leq D'_x(0)$, then $D'_i(p) \leq D'_i(0)$ for all states s_i with $i \neq x$, so let us focus on showing that $D'_x(p) \leq D'_x(0)$. We will do this by showing that the derivative of $D'_x(p)$ with respect to p is negative for all relevant p. In fact, by showing that this derivative is strictly negative, we establish the strict inequality of the lemma.

Now

$$\frac{d\,D'_x(p)}{d\,p} = \frac{N(p)}{(f(p))^2},$$

so it is enough to establish that $N(p) < 0$, where by equations (3.17) and (3.18)

$$
\begin{aligned}
N(p) &= D_y^{x,0}\, f(p) - \left(1 + \sum_{j=0}^{n} p_{xj}\, D_j^{x,0} \;+\; p\, D_y^{x,0}\right) \frac{d\,f(p)}{d\,p} \\
&= D_y^{x,0} \left(\sum_{j=0}^{n} p_{xj}\, g_j \;+\; p\, g_y\right) - \left(1 + \sum_{j=0}^{n} p_{xj}\, D_j^{x,0} \;+\; p\, D_y^{x,0}\right) g_y \\
&= D_y^{x,0} \left(\sum_{j=0}^{n} p_{xj}\, g_j\right) - g_y \left(1 + \sum_{j=0}^{n} p_{xj}\, D_j^{x,0}\right).
\end{aligned}
$$

The assumption of the lemma that $D_x > D_y$ says that

$$
\begin{aligned}
D_x &> D_y \\
&= g_y\, D_y^0 + (1 - g_y)\,[D_y^x + D_x] \\
&= D_y^{x,0} + (1 - g_y)\, D_x.
\end{aligned}
$$

(3.19) So $g_y D_x > D_y^{x,0}$.

From the expression for D_x given by equation (3.17) and the expression for $f(p)$ given by equation (3.18), we see that for $p = 0$

$$D_x = \frac{1 + \sum_{j=0}^{n} p_{xj} D_j^{x,0}}{\sum_{j=0}^{n} p_{xj} g_j}.$$

Thus equation (3.19) becomes

$$g_y \left(1 + \sum_{j=0}^{n} p_{xj} D_j^{x,0}\right) > D_y^{x,0} \left(\sum_{j=0}^{n} p_{xj} g_j\right).$$

But this says precisely that $N(p) < 0$, thereby establishing the lemma. ∎

Comments on Lemma 3.11

Lemma 3.11 nearly allows us to reverse the process described by lemma 3.10. The main difference is that lemma 3.10 refers to states by their labels, while lemma 3.11 refers to states by their expected convergence times. For a single state s_x and a single state s_y, as in the statement of lemma 3.11, this poses no serious problems. However, in order to prove claim 3.7 we will need to apply lemma 3.11 to several states s_x and several states s_y simultaneously. The following comments are intended to prove that the more general formulation of lemma 3.11 is valid.

Comment 1. Observe that if α is strictly negative in lemma 3.10, then none of the probabilities $\{p_{ij}\}$ need to become zero when they are changed, unless they are zero already. This means that a Markov chain that satisfies the probabilistic connectivity assumption of section 3.2.7 will continue to satisfy that connectivity condition after the modifications of lemma 3.10 have been performed. In other words, the goal reachability assumption of lemma 3.11 continues to be satisfied. The purpose of that assumption in the hypotheses of the lemma is simply to ensure that the expected convergence times are well-defined. In particular, the theory of Markov chains tells us that the system of linear equations relating those expected convergence times has a solution, and that that solution is unique.

Comment 2. Observe further that lemma 3.11 continues to hold when the single target state s_y is replaced by a multitude of such states. In particular, suppose that one is given a state s_x and a collection of states $Y = \{s_{y_1}, \ldots, s_{y_k}\}$ disjoint from s_x. Suppose further that there exist k non-negative numbers $\{\lambda_{y_1}, \ldots, \lambda_{y_k}\}$, that satisfy the conditions

$$\sum_{s_y \in Y} \lambda_y = 1, \qquad D_x > \sum_{s_y \in Y} \lambda_y D_y.$$

Then the conclusion of lemma 3.11 continues to hold, assuming that one takes $p'_{xy} = p_{xy} + \lambda_y p$, for each state $s_y \in Y$. The proof of the lemma goes through as before, except that g_y is replaced by $\sum_{s_y \in Y} g_y$ and D_y is replaced by $\sum_{s_y \in Y} \lambda_y D_y$.

Comment 3. If we look carefully at the proof of lemma 3.11, we see that the claim of the lemma can be considerably strengthened. Recall comment 2. Now let p increase from 0 to p_{xx}. Let us examine the behavior of the expected convergence times $\{D'_i\}_{i=0}^n$. We have a three-way case statement:

1. If $D_x > \sum_{s_y \in Y} \lambda_y D_y$, then D'_x decreases strictly, while all other D'_i either remain constant or decrease.

2. If $D_x < \sum_{s_y \in Y} \lambda_y D_y$, then D'_x increases strictly, while all other D'_i either remain constant or increase.

3. If $D_x = \sum_{s_y \in Y} \lambda_y D_y$, then all D'_i remain constant.

These comments follow from equation (3.16) and the computation of the derivative dD'_x/dp on page 135.

Comment 4. Finally, suppose that instead of a single state s_x one is given a set of states $\{s_x\}$ in lemma 3.11. Assume that for each such s_x there is a collection of states Y_x, whose weighted expected convergence times are less than the expected convergence time D_x of s_x, as outlined in comment 2. The claim is that if one simultaneously modifies the transition probabilities as outlined in comment 2 for each of the states s_x, then the expected convergence times of the resulting Markov chain improve.

We would like to know that this generalization of lemma 3.11 is correct. Ideally, in proving this generalization, one would iteratively apply lemma 3.11 and comment 2 for each of the states s_x in turn, until all the modifications suggested had been accomplished. Unfortunately, such an iterative application of lemma 3.11 need not be valid. This is because the lemma says little about the relative improvement in expected convergence times for different states. Thus, in performing the modifications suggested for one state s_{x_1}, one could possibly modify the expected convergence times of all the other states in such a way that the hypotheses of the lemma no longer are satisfied for some other state s_{x_2}. In particular, it could happen that

$$D'_{x_2} < \sum_{s_y \in Y_{x_2}} \lambda_y D'_y,$$

in the modified Markov chain. In that case, lemma 3.11 no longer applies. We thus need a slightly more elaborate argument. In particular, we will apply lemma 3.11 repeatedly. However, we will allow for the possibility that the transitions out of each state s_{x_i} may need to be modified several times.

Let us set up the general version of lemma 3.11, then offer a proof. The most general version simply assumes that one may change the transition probabilities at any non-goal state. Thus, for each non-goal state s_i in the state space, suppose

that we are given $n+1$ numbers $\{\lambda_{ij}\}_{j=0}^{n}$ that are either all zero (meaning that no transitions out of s_i are to be changed), or that satisfy the following four conditions:

$$\lambda_{ii} = 0, \qquad \lambda_{ij} \geq 0,\ j = 0, \ldots, n,$$

$$\textstyle\sum_{j=0}^{n} \lambda_{ij} = 1, \qquad D_i > \sum_{j=0}^{n} \lambda_{ij} D_j.$$

Now, for each state s_i for which the $\{\lambda_{ij}\}_{j=0}^{n}$ are not all zero, let q_i be any number satisfying $0 \leq q_i \leq p_{ii}$. Take q_i to be zero if all the $\{\lambda_{ij}\}_{j=0}^{n}$ are zero. The probabilities $\{q_i\}$ play the role of the probability p in the statement of lemma 3.11. Suppose that one constructs a new Markov chain on the old state space by modifying the probabilities as follows (for $i = 1, \ldots, n$):

$$p'_{ij}(q_1, \cdots, q_n) = \begin{cases} p_{ii} - q_i, & \text{if } j = i \\ p_{ij} + \lambda_{ij} q_i, & \text{otherwise.} \end{cases}$$

[We assume, as always, that there are no transitions out of the goal.]

Then the claim is that the expected convergence times $D'_i = D'_i(q_1, \cdots, q_n)$ of the new Markov chain are no worse than those of the old chain, for all legal values of $q_1, \ldots, q_n$.

Let us focus only on those q_i for which not all of the $\{\lambda_{ij}\}_{j=0}^{n}$ are zero, leaving all the other q_i fixed at zero. Consider one such q_i for a moment. The proof of lemma 3.11, in conjunction with comment 2, tells us that for each state s_j the expected convergence time $D'_j(0, \cdots, 0, q_i, 0, \cdots, 0)$ improves or remains the same as q_i varies from 0 to p_{ii}. However, the lemma says nothing about what happens if we vary several of the $\{q_i\}$ simultaneously. Indeed, it is not difficult to construct examples for which the expected convergence times first improve, then begin to get worse again, as the $\{q_i\}$ are each in turn increased from zero to their appropriate maximum values (p_{ii} for q_i). However, in all of these examples, the expected convergence times are always better than the initial expected convergence times of the unmodified chain. In other words, for all legal values of the $\{q_i\}$, and all states s_j, $D'_j(q_1, \cdots, q_n) \leq D'_j(0, \cdots, 0)$. We now outline a proof of this fact.

First, some notation. We will let the vector $\mathbf{q} \in \Re^n$ be shorthand for $(q_1, \cdots, q_n)$. Also, $\mathbf{q_{max}}$ will denote the vector $\mathbf{q}$ for which each of the q_i is at its maximum legal value (either 0 or p_{ii}). Finally, let $D'_i(\mathbf{q})$ denote the expected convergence time for state s_i, as determined by $\mathbf{q}$.

We will now construct a sequence $\mathbf{q_0}, \mathbf{q_1}, \ldots, \mathbf{q_m}$, such that $\mathbf{q_0} = \mathbf{0}$ and $\mathbf{q_m} = \mathbf{q_{max}}$. Furthermore, any one element $\mathbf{q_k}$ in this sequence differs from the previous element $\mathbf{q_{k-1}}$ in exactly one coordinate. In other words, $\mathbf{q_k} - \mathbf{q_{k-1}} = (0, \ldots 0, \Delta q_i, 0, \ldots, 0)$, for some $\Delta q_i > 0$, representing an increase in the value of q_i. The sequence $\mathbf{q_0}, \ldots, \mathbf{q_m}$ will be chosen in such a manner that as $\mathbf{q}$ varies from $\mathbf{q_k}$ to $\mathbf{q_{k+1}}$ the expected convergence times $\{D'_j(\mathbf{q})\}_{j=0}^{n}$ all either decrease or remain the same. It follows that

$D'_j(\mathbf{q_{max}}) \leq D_j$, for each $j = 0, 1, \ldots n$. And thus the general version of lemma 3.11 will be proven.

In order to construct the sequence $\mathbf{q}_0, \ldots, \mathbf{q}_m$, define the functions $L_i(\mathbf{q}) = -D'_i(\mathbf{q}) + \sum_{j=0}^{n} \lambda_{ij} D'_j(\mathbf{q})$, for $i = 1, \ldots, n$. Observe, by comment 3, that whenever $L_i(\mathbf{q})$ is negative, then q_i may be increased up to its maximum legal value without increasing any of the expected convergence times $D'_j(\mathbf{q})$. Furthermore, if actually $L_i(\mathbf{q})$ is zero, then q_i may be changed within its legal limits without affecting the expected convergence times at all.

By hypothesis, all the $\{L_i\}$ are negative or zero at $\mathbf{q} = \mathbf{0}$. Without loss of generality, we may assume that L_1 is negative. Then one may construct $\mathbf{q}_1$ by changing q_1. In particular, if it is possible to change q_1 to its maximum value without causing any of the $\{L_j\}$ to become positive, then we will do so. Otherwise, one of the $\{L_j\}$ must become zero for some value of Δq_1. In particular, suppose that $L_{j_1}(\mathbf{q}_1) = 0$ for $\mathbf{q}_1 = (\Delta q_1, 0, \ldots, 0)$. Then we can next allow q_{j_1} to vary from zero to its maximum legal value (say $p_{j_1 j_1}$), without affecting any of the convergence times. In other words, $\mathbf{q}_2 = (\Delta q_1, 0, \ldots, 0, p_{j_1 j_1}, 0, \ldots, 0)$. In computing $\mathbf{q}_3$ we again increase q_1. This is legal, since $L_1(\mathbf{q}_2) < 0$ by construction. We repeat this process, until q_1 has been increased to its maximum value, at which point we move on to some other q_i. The whole process is repeated several times, until all the q_i have been changed from zero to their maximum values.

The key observation in the process described above, is that we always modify only one q_i at a time, namely one for which $L_i \leq 0$. In particular, if $L_i = 0$, we modify q_i completely, and thereafter forget about it. If it is merely true that $L_i < 0$, then we are careful to modify q_i only so far until one of the other $\{L_j\}$ that are still under consideration becomes zero. And so forth.

We can now finally prove the main claim, namely Claim 3.7.

Proof of Claim 3.7. Recall that v is the maximum expected velocity of the Markov chain relative to the labelling $\{\ell_i\}$, and that this expected velocity is strictly negative. By Lemma 3.10, one can modify the transition probabilities so that the expected velocity at each state is exactly equal to v. Furthermore, for a given state s_x, the only changes to the transition probabilities entail increasing p_{xx} and decreasing p_{xy} for values of y that satisfy $\ell_y < \ell_x$. By Corollary 3.9 the expected success times of the resulting chain are precisely proportional to the state labels, with proportionality constant $-1/v$.

Now imagine reversing the modifications, so that one gets back the original chain. This is just the process described by Lemma 3.11 and subsequent comments. These observations therefore establish that the expected success times of the original chain are no greater than the success times of the modified chain. In short, $D_i \leq -\ell_i/v$, as claimed. ∎

3.6 Progress in Tasks with Non-Deterministic Actions

The claims of the previous section apply to actions in which the effects at each step and in each state are probabilistically determined. However, as we mentioned at the outset, for many tasks the actions are merely non-deterministic, that is, no probability distributions are given. In this case, the claims do not apply. In particular, the definition of an average velocity no longer makes sense. Of course, one can define a worst-case velocity, which measures the least amount of progress possible in any given state, and then variants of some of the claims will go through. This does not seem very satisfying, for two reasons. First, insisting that the worst-case velocity point towards the goal is not much different from insisting on deterministic actions. And second, often it is simply not possible to ensure that progress is made towards the goal. This is particularly true in the imperfect sensing case.

Surprisingly enough, however, the condition that the worst-case velocities point towards the goal characterizes the tasks for which solutions exist, at least when sensing is perfect. This section therefore presents a brief exposition of this condition.

First, we have the following claim.

Claim 3.12 *Let $(\mathcal{S}, \mathcal{A}, \Xi, \mathcal{G})$ be a discrete planning problem, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, Ξ is the sensing function, and $\mathcal{G}$ is the set of goal states. Assume that Ξ is the perfect-sensing function (this will be relaxed later).*

Suppose that there exists a guaranteed strategy for moving from any state to the goal set $\mathcal{G}$. Then there exists a sequence of disjoint sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_\ell$ that cover $\mathcal{S}$, such that states in the set $\mathcal{S}_{i+1}$ can traverse to states in the union $\mathcal{D}_i = \bigcup_{j=0}^{i} \mathcal{S}_j$ in a single step. Furthermore, $\mathcal{S}_0 = \mathcal{G}$, and $\ell \leq r = |\mathcal{S}| - |\mathcal{G}|$.

It follows that there is a perfect-sensing strategy that moves through the tower of sets $\mathcal{S} = \mathcal{D}_\ell \supset \cdots \supset \mathcal{D}_1 \supset \mathcal{D}_0 = \mathcal{G}$ in decreasing order until the goal is attained. Furthermore, the strategy moves down at least one level in this tower on each step of execution.

[A definitional note: To say that a strategy is in one of the levels $\mathcal{D}_i$ at a particular time, means that at that time one of the possible states of the system is in the set $\mathcal{S}_i$ and no state is in a set $\mathcal{S}_j$, with $j > i$. To say that a strategy moves between two levels means that there is an execution trace for which the strategy first finds itself in one level, then one step later in the next level.]

Proof. The proof is based on the construction used in the proof of claim 3.4, which we repeat here. Define $\mathcal{S}_0$ to be $\mathcal{G}$, and then inductively define $\mathcal{S}_{i+1}$ to be the set of all states s in $\mathcal{S} - \bigcup_{j=0}^{i} \mathcal{S}_j$ for which there exists a single-step action that causes s to traverse to some state in the set $\bigcup_{j=0}^{i} \mathcal{S}_j$. The sets $\mathcal{S}_i$ are well-defined, and by construction they are disjoint. We need to show that they cover $\mathcal{S}$ and that there are no more than r of them. Note that the set $\bigcup_{j=0}^{i} \mathcal{S}_j$ is just the set of all states that can be guaranteed to reach a goal state in i or fewer steps. The existence of a guaranteed

strategy means that all states can be guaranteed to reach the goal in a finite number of steps, so the $\{S_i\}$ cover S. Finally, by claim 3.5, no more than r steps are ever required.

The second part of the claim follows immediately. ∎

This claim is very similar to the proofs of claims 3.4 and 3.5. The difference is that the current claim emphasizes the overall structure of the perfect-sensing strategy. Not only do individual execution-time traces of the perfect-sensing strategy never need to revisit a state, but seen as a whole, the strategy should permanently prune away possible states on each step. Intuitively this makes sense. After all, if there is some state that does not get pruned away, then it is possible to repeatedly encounter that state, which means that the strategy cannot be guaranteed to converge to the goal.

Notice also that the fact that the sensing function is perfect is really not used in the proof, except to limit the number of sets that are required. This should not be surprising, given the equivalence between the existence of a perfect-sensing strategy and goal reachability in general, as established by claim 3.4. This then leads us to believe that the claim holds for an arbitrary sensing function, and indeed it does, with precisely the same sets $\{S_i\}$. However, whereas these sets actually define a strategy in the perfect-sensing case, they only hint at one in the general case. Specifically, one has the following corollary, which despite the length of statement, is actually quite weak.

Corollary 3.13 *Consider a system as in the previous claim, but with an arbitrary sensing function Ξ. Construct the sets $\{\mathcal{D}_i\}$, as in the proof of the claim. Suppose there is a strategy that is guaranteed for each possible starting state to attain the goal set $\mathcal{G}$.*

Then this strategy must necessarily move through the sets $\mathcal{S} = \mathcal{D}_\ell \supset \cdots \supset \mathcal{D}_1 \supset \mathcal{D}_0 = \mathcal{G}$ in decreasing order until the goal is attained. However, the strategy can spend several steps of execution within one level of this tower before proceeding down to a lower level, and can even move back up levels.

Proof. If the strategy is in level D_i, then there is at least one possible state of the system that may require i steps to reach the goal. This says that there is a possible execution trace for which the system must first pass through an immediately lower level before reaching the goal. ∎

In order to summarize, suppose we have a non-negative labelling of the states $\{\ell_i\}$ that is zero at the goal. Now define for any state s and any action A, the worst-case velocity to be

$$v_{A,s} = \max_{t \in F_A(s)} (\ell_t - \ell_s),$$

where $F_A(s)$ is the the set of states that the non-deterministic action A might transit to from state s.

In a sense, this velocity measures the minimum approach to the goal, relative to the labelling. If $v_{A,s}$ is negative, then no matter which non-deterministic transition is actually followed, the system is making progress.

For the perfect-sensing case, we see that one can label all states in the set S_i with the number i. Then the worst-case velocity at each point is -1, and the system will reach the goal in no more than $i = -\ell_{s_k}/v_{s_k,A_k}$ steps, for each state s_k and its perfect-sensing action A_k. In short the formalism carries through, trivially, in the perfect-sensing case.

In the more general case, it is not clear whether it really makes sense to define the worst-case velocity at a state. For one thing, the action executed in a state is not well-defined, since a state may be revisited several times during the execution of a strategy, but the action executed will generally depend on the system's knowledge state, which will be different. Hand in hand with this is the lack of substance provided by a progress measure on the state space, as indicated by corollary 3.13. However, if one not only maximized over all possible target states in the definition, but also over all possible actions that a strategy might execute in a given state, that is, if one defined the worst-case velocity to be

$$v_s = \max_{\substack{\text{applicable} \\ \text{actions } A}} \max_{t \in F_A(s)} (\ell_t - \ell_s), \tag{3.20}$$

then the linear convergence result basically goes through as before. This result is probably too weak to be useful.

One issue may be bothersome. In the probabilistic setting, this strong distinction regarding the use of a progress measure with perfect sensing versus the use of a progress measure with imperfect sensing did not seem to arise. In fact, it does arise, but this was not relevant to the discussion on Markov chains. In the discussion of progress measures on Markov chains we were tacitly assuming that the effect of an action and the interpretation of a sensor in deciding on an action depended only on the current state of the system. This makes sense in the perfect-sensing case. It also makes sense if actions are selected directly as a function of sensor values, and not as a function of the interpretation of those sensor values in terms of past information. In general, however, the interpretation of a sensor depends on the knowledge state of the executive system, not just on the actual state of the system (see sections 3.2.5 and 3.2.6). Once one re-introduces this dependence, then the distinction between perfect and poor sensing becomes important, both in the non-deterministic and the probabilistic settings.

3.7 Imperfect Sensing

In general, given an imperfect sensor, the appropriate states of the system are the knowledge states. In the non-deterministic setting these are all the subsets of the underlying state space, while in the probabilistic setting these are all the probability distributions over the underlying state space. One can then define labellings and

progress measures as before on this space of knowledge states, and all the results will go through. Formally, this is the correct description of the problem. However, as we have already indicated, the general planning problem is hard, which is reflected in the exponential size of the space of knowledge states. For this reason one seeks less complete approaches that nonetheless can handle a variety of tasks. This is precisely the reason that we decided to consider the special case settings of random walks and Markov chains in the first place.

3.8 Planning with General Knowledge States

In order to deal with the general sensing case, it is useful to consider a planner for determining guaranteed strategies for achieving the goal. A guaranteed strategy in this context means a bounded number of actions and sensory operations that are certain to attain a goal state from the specified initial states, under the specified model of uncertainty. In general the actions will be functions of sensors, that is, the strategy will involve conditional choices based on non-deterministic events whose outcomes cannot be predicted at planning time. Nonetheless, the flow graph of these choices and non-deterministic events can be written out, and it has finite size and converges to the goal. The planning approach described in this section is a specialization of the *preimage* planning approach defined by [LMT]. It may also be thought of as dynamic programming with a boolean cost function (see [Bert]). Before reading this section, it may be worthwhile to reread the example of section 3.2.4.

The basic idea is to apply backchaining in a state-space whose states are the knowledge states of the executive system. By construction, "sensing" in such a space is perfect. This means that by definition the system knows exactly which (knowledge) state it is in at any point during execution. The approach is applicable to both the non-deterministic and the probabilistic settings. Let us just briefly outline how the planner might proceed for the non-deterministic setting. If $\mathcal{S}$ is the underlying state space, then the planner's state space is the set of all knowledge states, that is, the space $2^{\mathcal{S}}$. Let us assume that an action is always followed by some kind of sensing operation (possibly a no-op). If A is an action in the underlying state-space, and K_1 is a knowledge state, then the result of applying action A is a new knowledge state K_2 (see section 3.2.5 for the details of how to construct K_2). Now suppose that a finite number of sensory interpretation sets can be returned by the sensor after the action has been executed. The actual sensory interpretation set will in general depend on the actual state of the system, plus possibly several other parameters. Let this collection be $\{I_1, I_2, \cdots, I_\ell\} = \bigcup_{s \in K_2} \Xi(s)$, and define K_2^i by $K_2^i = K_2 \bigcap I_i$. Then we can write the non-deterministic effect of the combination of action A and sensing in the space $2^{\mathcal{S}}$ as

$$\hat{A}: \quad K_1 \mapsto \quad K_2^1, K_2^2, \ldots, K_2^\ell$$
$$\vdots \qquad \vdots$$

This means that at execution time the action $\hat{A}$ (which corresponds to performing action A followed by some sensory operation) will transit non-deterministically from (knowledge) state K_1 to one of the states K_2^i. By construction, our sensing guarantees that the execution system will know precisely which knowledge state has been attained. Thus the problem is a perfect-sensing problem.

Since the problem has a perfect-sensing function, one can apply the techniques previously discussed for such problems. In particular, one can plan strategies for achieving a goal state (and knowing that it has been achieved) by backchaining from the goal in the space 2^S. This amounts to applying the dynamic programming discussed in section 3.2.4. Backchaining entails first determining the collection $\mathcal{K}_1$ of all knowledge states that can attain a goal state with the execution of a single action-sense pair, then determining the collection $\mathcal{K}_2$ of all knowledge states that can attain one of the knowledge states in the collection $\mathcal{K}_1$ using a single action-sense pair, and so forth. This construction is identical to the construction of the sets $\{\mathcal{S}_i\}$ in claim 3.12, but now these sets reside in the space 2^S. The method of transforming an imperfect-sensing problem into a perfect-sensing problem by moving to the space of knowledge states is a standard technique (see, for instance, [Bert]). As we see, this transformation combines an action and a sensory operation in the underlying state space into a perfect-sensing operation in the space of knowledge states. An alternate approach is to model the effect of actions and sensing operations as defining an AND/OR graph in the knowledge space (see [Buc] or [TMG] for details).

There are two basic possible formulations of the planning problem. One is to seek a sequence of motions that is guaranteed to move the initial knowledge state $\mathcal{I}$ to the goal state $\mathcal{G}$. [The notation may be confusing, since so far we have thought of $\mathcal{G}$ as being a set of goal states, but in the space of knowledge states this is just one state. More generally, one could have several such sets $\{\mathcal{G}_i\}$, and the formalism would go through.] A second approach is to limit *a priori* the number of steps considered. In this case, one backchains for the specified number of steps, whereas in the previous case one backchains until all knowledge states have been considered. In both cases, of course, one can stop if at any step the backchaining process generates the knowledge state $\mathcal{I}$.

An example should clarify all this notation:

Suppose there are three states, s_1, s_2, and s_G, where s_G is the goal state. Suppose that our sensor is good enough to tell us whether we are in the goal or not, but cannot distinguish between states s_1 and s_2. Finally let there be two actions, A_1 and A_2, specified by:

$$A_1: \begin{array}{l} s_1 \mapsto s_2, s_G \\ s_2 \mapsto s_2 \\ s_G \mapsto s_G, \end{array} \qquad A_2: \begin{array}{l} s_1 \mapsto s_1 \\ s_2 \mapsto s_G \\ s_G \mapsto s_G. \end{array}$$

These actions are depicted in figure 3.14.

The space of knowledge states is given by the seven sets (we exclude the empty set, which implies inconsistent knowledge):

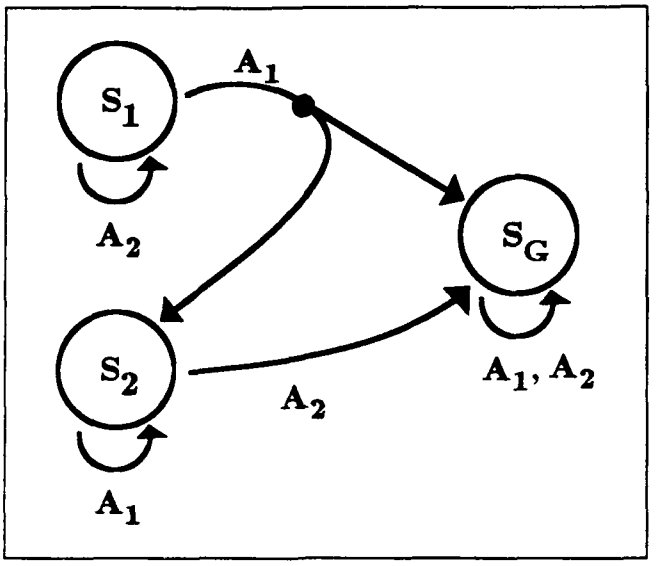


Figure 3.14: Some simple non-deterministic actions on a discrete state graph.

$$\{s_1, s_2, s_G\}, \{s_1, s_2\}, \{s_1, s_G\}, \{s_2, s_G\}, \{s_1\}, \{s_2\}, \{s_G\}.$$

For instance, the knowledge state

$$\{s_1, s_2\}$$

means that at execution time the system knows that it is either in state s_1 or state s_2, but it does not know which one. If the system always performs a sensory operation after each action, then, since the sensor can recognize the goal state for sure, one may actually eliminate from the space of knowledge states all states that contain both the goal state and some other state. Thus the relevant planning space is given by the four knowledge states

$$\{s_1, s_2\}, \{s_1\}, \{s_2\}, \{s_G\},$$

with $\mathcal{G} = \{s_G\}$ as goal state.

Let us compute the actions induced in the knowledge space by an action-sense pair. For action A_1 we have:

$$\hat{A}_1: \begin{array}{rcl} \{s_1, s_2\} & \mapsto & \{s_2\}, \{s_G\} \\ \{s_1\} & \mapsto & \{s_2\}, \{s_G\} \\ \{s_2\} & \mapsto & \{s_2\} \\ \{s_G\} & \mapsto & \{s_G\}, \end{array}$$

while A_2 becomes:

$$\hat{A}_2: \begin{array}{rcl} \{s_1, s_2\} & \mapsto & \{s_1\}, \{s_G\} \\ \{s_1\} & \mapsto & \{s_1\} \\ \{s_2\} & \mapsto & \{s_G\} \\ \{s_G\} & \mapsto & \{s_G\}. \end{array}$$

See figure 3.15 for a graphical display.

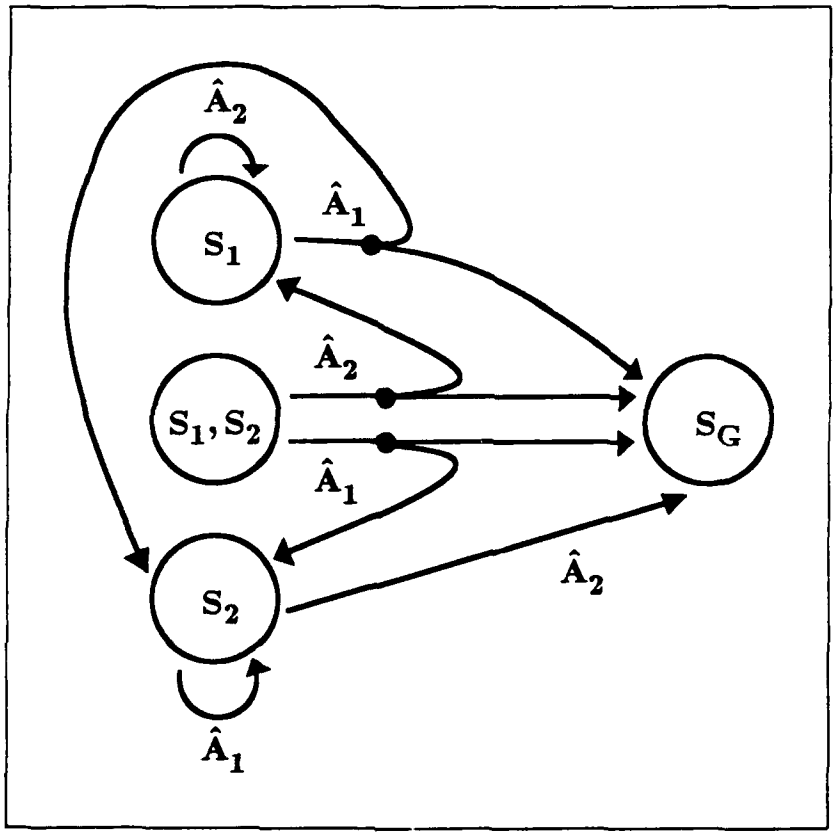


Figure 3.15: This figure displays the knowledge states and actions corresponding to the diagram of figure 3.14.

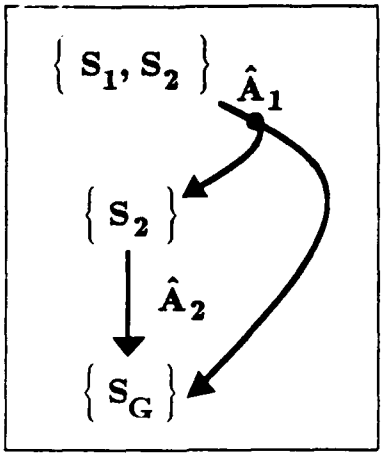


Figure 3.16: This figure shows how a backchaining strategy might evolve in the space of knowledge states. See also figure 3.15.

Here is the dynamic programming table for this problem. The horizontal axis of the table indicates the number of steps remaining, the vertical axis indicates the current knowledge state. Each entry indicates the action to take in order to attain the goal, given the number of steps remaining and the current knowledge state. The table is constructed by first backchaining from the goal, in order to construct all entries in the column for one remaining step, then backchaining from that column, and so forth. A blank entry indicates that it is not possible to successfully and recognizably attain the goal in the number of steps specified from that knowledge state.

Steps Remaining			
2	1	0	
$\hat{A}_1$			$\{s_1, s_2\}$
$\hat{A}_1$			$\{s_1\}$
$\hat{A}_2$	$\hat{A}_2$		$\{s_2\}$
stop	stop	stop	$\{s_G\}$

Knowledge States

Actions guaranteed to attain the goal.

As the table indicates, it is possible to attain the goal from any non-goal initial state of knowledge in at most two steps.

Let us relate this notation to the preimage planning methodology developed by [LMT] (see also chapter 4). The entry in column 1 says that the preimage of the goal under action $\hat{A}_2$ is the set $\{s_2\}$. This means that if the system knows that it is in state s_2, then executing action A_2 followed by a sensing operation is guaranteed to attain the goal. This is written as $P_{\hat{A}_2,R}(\{s_G\}) = R$, for $R = \{s_2\}$. Similarly, the top entry of column 2 comes from the fact that the set $\{s_1, s_2\}$ is the preimage under

action $\hat{A}_1$ of the two sub-goals $\{s_2\}$ and $\{s_G\}$. In the preimage methodology this is written as $P_{\hat{A}_1,R}(\{s_2\},\{s_G\}) = R$, for $R = \{s_1, s_2\}$. This means that if the system starts out knowing only that it is in either state s_1 or state s_2, then after action A_2 and a sensing operation, the system will have traversed to either state s_2 or the goal s_G, *and* the system will know which state it has attained. Figure 3.16 depicts this graphically.

3.9 Randomization with Non-Deterministic Actions

As we have formulated the problem thus far, the planner constructs a circuit of knowledge states by backchaining from the goal. The problem is considered solved if one of these knowledge states contains the initial state of the system. This is what is meant by a guaranteed solution throughout this thesis. For some tasks however, there is no such guaranteed solution. We mentioned this in the introduction. We will quickly review the example of figure 1.17 on page 42 from the introduction. Assume that the goal is recognizable, that is, that the sensing function for this problem can detect entry into the goal state. If the initial state of the system is known exactly, then there is a simple solution for attaining the goal. Specifically, if the system is in state s_1, then execute action A_1, whereas if the system is in state s_2, then execute action A_2. On the other hand, if the initial knowledge state is the set $\{s_1, s_2\}$ then there is no sequence of actions guaranteed to attain the goal. Fortunately, there exists a randomized solution that is expected to attain the goal very quickly. This solution consists of guessing the state of the system, then executing the action appropriate for that state. In this simple example, there are two possible choices for the starting state. Thus, with probability 1/2 the randomized strategy will guess the correct starting state. It follows that the expected time until goal attainment is two attempts.

This same approach of randomizing the initial state may of course be applied even if there exists a guaranteed solution. The motivation would be to find a randomized solution that requires fewer steps on average than the guaranteed solution.

3.9.1 Guessing the Starting State

Let us specify formally the relationship of randomization by guessing with the guaranteed planning approach. As usual, we will view the planning process in terms of backchaining, and specifically, in terms of dynamic programming in the space of knowledge states. Consider the column in the dynamic programming table that corresponds to i steps remaining in the strategy. Consider all the knowledge states $\{K_{i,1}, K_{i,2}, \cdots, K_{i,\ell}\}$ in this column that have non-blank entries. These are all the knowledge states for which there exists a sequence of at most i actions guaranteed to attain the goal. This collection is precisely the set $\mathcal{D}_i$, in the notation of claim 3.12. Suppose that $\mathcal{I}_0$ is the initial knowledge state of the system. If $\mathcal{I}_0 = K_{i,j}$ for some j,

then there is a guaranteed strategy consisting of no more than i steps that will attain the goal. More generally, however, we may have that

$$\mathcal{I}_0 \subseteq \bigcup_{j=1}^{\ell_i} K_{i,j}.$$

In that case there exists a randomized strategy that consists of guessing an effective knowledge state. To see this, suppose that $\mathcal{I}_0$ is of the form $\{s_1, s_2, \cdots, s_q\}$. In other words, there are q (with $q \leq n = |\mathcal{S}|$) possible starting states of the system. Thus there exist q or fewer knowledge states in the collection $\mathcal{D}_i$ which cover $\mathcal{I}_0$. We may thus assume that

$$\mathcal{I}_0 \subseteq \bigcup_{j=1}^{q} K_{i,j}$$

A randomized strategy consists of guessing between these q knowledge states, then executing the proper sequence of actions designed to attain the goal. For instance, if the system were to guess that $K_{i,j}$ is a knowledge state that contains the actual starting state of the system, then henceforth the system would execute all actions and sensing operations as if the initial knowledge state really had been $K_{i,j}$ instead of $\mathcal{I}_0$. Since the states $\{K_{i,1}, K_{i,2}, \cdots, K_{i,q}\}$ cover $\mathcal{I}_0$, the guess will be correct with probability no less than $1/q$. Thus with probability at least $1/q$ the goal will be successfully attained with a strategy requiring i or fewer steps.

A simple example of this state-guessing approach is given by the two-dimensional peg-in-hole problem of figure 2.2. If the resolution of the sensor is not good enough to determine whether the peg is to the left or to the right of the hole, then a useful strategy is simply to guess the side on which the peg is located. Depending on the outcome of the guess, the system then moves either right or left. If the guess is correct, the the peg winds up in the hole. If the guess is incorrect, then the system can either guess again or use the failure as information to select the appropriate direction of motion.

A more complex example will be given in the continuous domain in chapter 4. See in particular figure 4.8.

3.9.2 Execution Traces

In order to gain some intuition as to the types of execution traces that might occur, let us consider a randomized system at execution time. The system first guesses its starting knowledge state, then executes some appropriate strategy. This strategy is a guaranteed strategy for attaining the goal, in the sense that the strategy would reliably and recognizably attain the goal if the system knew for certain its starting knowledge state. However, since the starting knowledge state is merely guessed, it is possible that the system may encounter an inconsistency at execution time, reflected by the empty knowledge state. We assume that the system ceases execution of its

current guessed strategy should it ever encounter the empty set as a knowledge state at run-time.

In general, the system might actually be able to decide that it has attained the goal, even though an inconsistency has occurred (see claim 3.14 below). This decision involves an additional test, that essentially takes into account the effect of all the past actions and sensory interpretations on the entire range of possible starting states, not just on those in the guessed knowledge state.

There are thus two different notions of failure to recognizably attain the goal. One notion refers to failure relative to the guessed starting region. This failure is evidenced either by the occurrence of an inconsistency or by the presence of non-goal states in the knowledge state derived from the guessed starting knowledge state. The other notion refers to failure of the more accurate test, which takes into account all possible starting states. No inconsistencies can occur here, and thus this failure is evidenced merely by the presence of non-goal states in the knowledge state derived from the initial starting region $\mathcal{I}_0$. Either notion of failure is reasonable, depending on whether the more accurate test is implemented.

Suppose that a failure, by either definition, does occur. Then, under suitable conditions, the system may guess a new starting knowledge state, execute a new strategy for the newly guessed knowledge state, and so forth, repeatedly, until the goal is finally attained. We will elaborate on these conditions shortly.

In short, there are a couple of subtleties that need to be addressed. The first issue deals with the behavior of the system if it guesses the wrong starting state. The second issue deals with repeated guessing.

3.9.3 Incorrect Guessing

First, consider the behavior of the system if it guesses the wrong starting state. There are four possible results: (1) The system completes execution without thinking that it has attained the goal (although it may have), (2) the system thinks that is has attained the goal when indeed it has, (3) the system thinks that it has attained the goal when in fact it has not, and (4) the system encounters an inconsistency during execution.

The first two of these scenarios are standard and do not require elaboration. As an aside, let us note that scenario number one does not actually occur in the current context. This is because the system is executing a strategy that is guaranteed to attain the goal state from the (incorrectly) guessed starting state. Thus, either an inconsistency must occur during execution, or the system must eventually believe that it has attained the goal.

In order to understand the other two possibilities, imagine the behavior of the system if it guesses a knowledge state $K_{i,j}$ that does not contain the actual initial state of the system. At each step, the system will perform some action and some sensing operation as specified by the dynamic programming table. This action and the returned sensed value are used to update the knowledge state, in the manner described in section 3.2.5. However, since the knowledge state at each step of execution may

not contain the actual state of the system, the resulting sensory interpretation sets may not be consistent with the predicted forward projection of the knowledge state. In other words, the set $K_2 = F_A(K_1) \bigcap I$ may be empty, where K_1 is some knowledge state, $F_A(K_1)$ is the forward projection of K_1 under some action A, and I is the result of some sensing operation. One sees then that under this randomized strategy, the empty set can appear as a knowledge state. If ever it does appear, then the system knows that it has guessed incorrectly, and that it should stop execution. In fact, inconsistencies can arise more generally, if the full sensing consistency requirement is not satisfied. At any sensing time, the system knows what the possible sensor values are that it should be able to see. If a different sensor value actually appears, then an inconsistency has occurred, and the system knows that it originally guessed incorrectly. Said differently, the set $F_A(K_1) \bigcap' I$ is empty (recall the meaning of $\bigcap'$ from section 3.2.5). This explains scenario number four.

Scenario number three can occur precisely when no inconsistencies appear, despite the initial guess having been wrong. In other words, the execution trace of knowledge states from some initial knowledge state $K_{i,j}$ to the goal $\mathcal{G}$ proceeds successfully, despite the system not being in $K_{i,j}$ initially. In some cases the system may wind up in $\mathcal{G}$ serendipitously, but this need not be guaranteed. An example is given in figure 3.17. In this example there are two possible starting positions. The action executed is to move straight down, until a collision with a horizontal edge is detected. There are two such edges, one of which is the goal. If the system guesses that it has started at the point p_2 (which lies above the goal edge), but is really at location p_1, then the knowledge state at the end of the motion will incorrectly indicate goal attainment.

See also [Don89] for further details on the implications of "lying" to a system at run-time by specifying the wrong start location. Donald has used this technique in his work on Error Detection and Recovery to suggest multi-step strategies for trying to attain some goal, in such a manner that the process winds up distinguishing between those start locations that are guaranteed to attain the goal and those that merely might attain the goal. Clearly the process of guessing the start region has strong connections to his approach, as will become apparent in this section.

3.9.4 Goal Recognizability

Of the four scenarios, the only troublesome one is this third, the problem of false goal recognition. The resolution of this problem requires an applicability condition. Essentially the idea is to eliminate all possible execution traces that could lead to confusing goal interpretations. Specifically, for any execution trace that does indicate goal attainment, we want to ensure that the same execution trace applied to other possible initial knowledge states either also indicates goal attainment or leads to an inconsistency. We will state this condition formally, then simply enforce it by assuming that the goal is recognizable independent of any history, that is, any particular execution trace.

In order to state the condition formally let us introduce some temporary notation. This discussion applies to the non-deterministic setting, but not necessarily to the

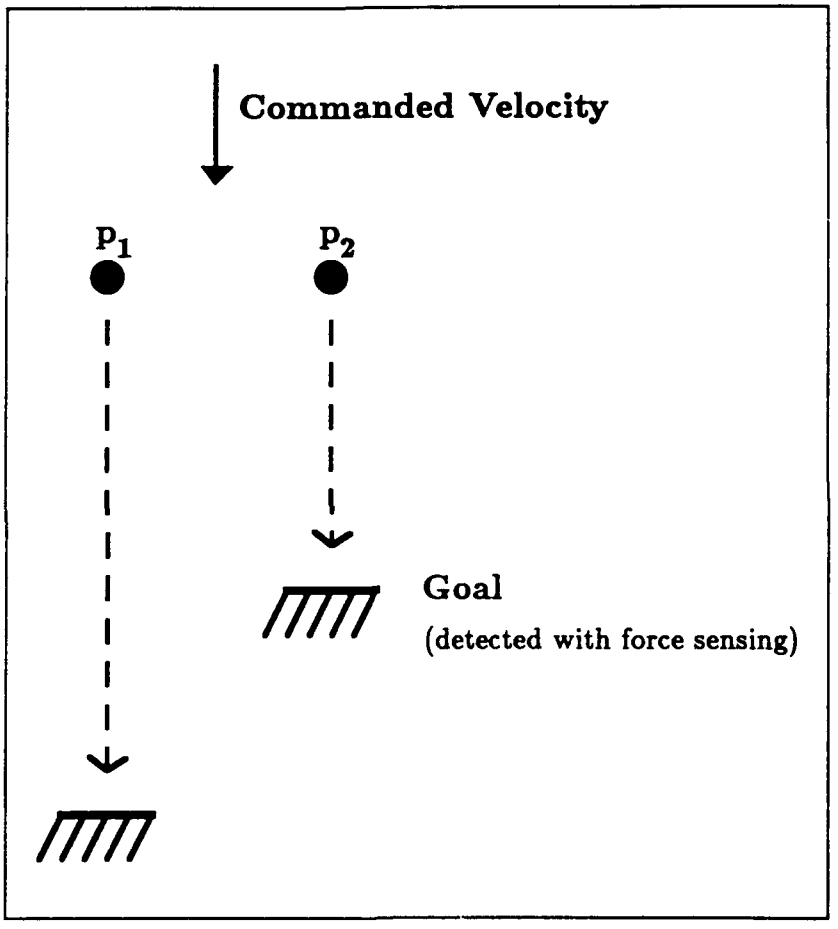


Figure 3.17: The system starts in one of the two indicated locations, moves downward, and detects contact with a horizontal surface. If the system *knows* that it started at location p_2, then the contact signals goal attainment. However, if the system merely *guessed* that it started at p_2, then the force sensor may falsely signal goal attainment.

probabilistic setting. Given a starting knowledge state K, and a non-deterministic action $\hat{A}$ in knowledge space, let us write the effect during execution of this action on K as $K; A; I$, where A is the generating non-deterministic action in the underlying state space, and I is a sensory interpretation set that is returned by the sensor at execution time. In other words, $K_2 = K; A; I$, where K_2 is the knowledge state determined from K in the manner of section 3.2.5, namely as $F_A(K) \bigcap' I$, the intersection of the sensory interpretation set with the forward projection of the start state. More generally, given a sequence of actions $\{A_1, A_2, \ldots, A_k\}$ and an associated sequence of run-time sensory interpretation sets $\{I_1, I_2, \ldots, I_k\}$, the effect on K will be denoted by $K; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$. If ever a sensory interpretation set is returned that is inconsistent with the possible sensory interpretation sets expected at that point, the resulting knowledge state is simply the empty set $\emptyset$. For consistency, we therefore define $\emptyset; A; I = \emptyset$ for any action A and any sensory interpretation set I.

Suppose now that the system guesses that the initial knowledge state is the set K_0. The strategy for attaining the goal G from K_0 is encoded in the dynamic programming table. Suppose that the first action, $\hat{A}_1$, is taken from the entry for K_0 in the k^{th} column of the dynamic programming table. Execution of $\hat{A}_1$ involves execution of some action A_1 on the underlying state space, followed by some sensory observation that yields a sensory interpretation set I_1. Once $\hat{A}_1$ has been executed, the resulting knowledge state determines the next action to perform. This action $\hat{A}_2$ is again encoded in the dynamic programming table. Action A_2 in turn results in some new run-time sensory interpretation set I_2, and so forth. If the initial state of the system was indeed covered by the starting knowledge state K_0, then after k actions the resulting knowledge state will be non-empty and inside the goal, that is, $\emptyset \neq K_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k \subseteq \mathcal{G}$. The precise sequence is of course not determined until execution time. On the other hand, if the initial state of the system was not covered by K_0, then the final knowledge state may or may not be empty, and may or may not accurately depict whether the goal has been attained, as explained above.

Now consider the effect of the sequence $A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ on knowledge states other than the assumed starting knowledge state K_0. In particular, consider $\{s_i\}; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ for all singleton knowledge state $\{s_i\}$. Suppose that for each possible starting state s_i, the final knowledge state $\{s_i\}; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ is either the emptyset $\emptyset$ or lies inside the goal $\mathcal{G}$. Then clearly the goal must have been attained, even if the initial guess K_0 was wrong! Conversely, suppose that for some state s_i, the final knowledge state is non-empty and includes states outside of the goal. If s_i could have been a starting state of the system, then one cannot be sure that the system has entered the goal. This establishes the following claim.

Claim 3.14 *Consider a discrete planning problem $(\mathcal{S}, \mathcal{A}, \Xi, \mathcal{G})$ for which the full sensing consistency requirement holds. Suppose the initial state of the system is known to lie in some subset $\mathcal{I}_0 \subseteq \mathcal{S}$. Suppose further that there exists a guaranteed strategy for attaining the goal in k steps if the initial state were actually known to be in the set K_0, with $K_0 \subseteq \mathcal{I}_0$. Imagine that the system executes this strategy as*

if the initial knowledge state were indeed K_0. Let the execution trace be given by $A_1; I_1; A_2; I_2; \cdots; A_k; I_k$. Then the system is guaranteed to have attained the goal if and only if $\mathcal{I}_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k \subseteq \mathcal{G}$.

[Notice that $K_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ may be the empty set, if the initial state of the system is not in K_0. However, the knowledge state $\mathcal{I}_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ must be non-empty, since the system is known to have started in the set $\mathcal{I}_0$, and since sensing is at least partially consistent.]

Proof. The claim follows from the discussion above, and the fact that

$$\bigcup_{s \in K} (\{s\}; A; I) = K; A; I,$$

for any knowledge state K, by lemmas 3.1, 3.2, and 3.3. ∎

As an aside, notice that the proof of the claim never made use of the fact that the execution trace was the result of executing a strategy guaranteed to move K_0 to the goal. This suggests that the claim holds for any strategy, and indeed it does, but this is not of use in this context.

Definition. Let us define the phrase *the strategy is assured of reliable goal recognition from* K_0 to mean that any execution trace of the strategy, which transforms K_0 into a non-empty knowledge state within the goal, actually implies goal attainment.

With the same hypotheses as the claim above, one obtains the following corollary. The corollary is merely a restatement of the definition of reliable goal recognition.

Corollary 3.15 *Suppose that a randomized strategy guesses that the system is in K_0, and plans to execute the guaranteed strategy for K_0, even though the actual state of the system may be in $\mathcal{I}_0 - K_0$. The strategy is assured of reliable goal recognition from K_0 if and only if $\mathcal{I}_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k \subseteq \mathcal{G}$ for all possible execution traces that might occur for which $\emptyset \neq K_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k \subseteq \mathcal{G}$.*

[Observe that the collection of possible execution traces is the union over all possible starting states in $\mathcal{I}_0$ of execution traces that might occur when executing the guaranteed strategy for K_0, not just the possible execution traces that might occur when executing the guaranteed strategy for K_0 knowing that the initial state is in K_0. However, the corollary only requires consideration of those execution traces that are consistent with K_0.]

The condition of this corollary forms the applicability condition for a randomized strategy. If the condition is satisfied for all possible knowledge states K_i that might be guessed, then false goal recognition is avoided.

As an aside, observe that if one does implement the more accurate test to determine whether $\mathcal{I}_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k \subseteq \mathcal{G}$, then corollary 3.15 is irrelevant. The corollary really tells us the conditions under which a local test relative to the guessed starting state K_0 is sufficient to ensure global goal attainment.

A couple of additional comments are in order. First, a quick reading of the corollary suggests that goal recognition is only reliable if the entire possible starting region $\mathcal{I}_0$ is guaranteed to attain the goal. If that were indeed true, all this discussion would be absurd, since one could simply apply the guaranteed strategy applicable to $\mathcal{I}_0$ rather than K_0. In fact, however, the corollary merely asserts that any execution trace starting from K_0, for which the final knowledge state derived from K_0 is non-empty and lies inside the goal, must also place the final knowledge state derived from $\mathcal{I}_0$ inside the goal. It is quite possible that on a particular execution trace the final knowledge state $K_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ is empty. In that case, the result of applying the strategy to $\mathcal{I}_0$ clearly need not achieve the goal. As we see from claim 3.14, the goal might actually be attained, but this is not guaranteed. Thus the randomized strategy would signal failure of its current attempt, based on the recognition that it had guessed wrong initially. In short, there need not be a guaranteed strategy for attaining the goal from $\mathcal{I}_0$.

The second comment concerns the relationship of the corollary to Donald's work on Error Detection and Recovery [Don89]. He, as we, was interested in executing a strategy from some large starting region, although the strategy was only guaranteed to attain the goal from some smaller subregion. The condition he placed on such a strategy was that it terminate by either recognizing goal attainment or recognizing attainment of a region from which goal attainment is impossible. The situation in our case is slightly different. In particular, as we shall see, the randomized strategy will actually loop over several attempts, on each making a new guess as to the effective starting state. After all, we have assumed that the large starting region is covered by a union of smaller regions, for each of which there exists a guaranteed strategy. This is a more stringent requirement than that Donald asked of his starting regions. Additionally, whereas Donald required his strategies to either recognize success or failure, we have simply defined failure to be the lack of success. Indeed, it may happen that the strategy terminates thinking it has failed when in fact the state of the system is inside the goal. Our only requirement is that if the strategy thinks that it has attained the goal, then indeed it has. This is a weaker requirement, one that is more easily satisfied. It is enough for our purposes, since on each iteration of the randomized strategy, there is some non-zero probability of guessing the correct starting state, and thus some non-zero probability of terminating successfully.

3.9.5 Repeated Goal Reachability

The second issue that needs to be addressed concerns the behavior of the randomized strategy upon failure.[3] Thus far we have merely asked that the strategy guess a starting knowledge state and execute a strategy guaranteed to achieve the goal if the guess is correct. If the guess is incorrect and the strategy fails to achieve the goal, then one needs to worry about how to proceed. One possibility is that the new resulting knowledge state at execution time is one of those for which a guaranteed strategy

[3] As before, failure can have two meanings, either relative to the guessed starting region, or relative to the entire starting region. Either meaning is acceptable. See section 3.9.2.

exists. In other words, a non-blank entry appears in the dynamic programming table for that knowledge state. Another possibility is that the new knowledge state is the union of several smaller knowledge states for which guaranteed strategies exist. More, generally, however, there may not be any way to proceed. This leads to a second applicability condition.

Consider a k-column dynamic programming table. Suppose that the initial state of the system is known to lie in some subset $\mathcal{I}_0$ of the state space. Assume as before, that there is a collection of at most $n = |\mathcal{S}|$ knowledge states that cover the set $\mathcal{I}_0$, from each of which there is a guaranteed strategy of k steps for attaining the goal. Now let us go one step further. Consider the i^{th} column of the table, and define $\hat{\mathcal{D}}_i$ (for $i = 1, \ldots, k$) to be the union of all knowledge states whose entries in this column are non-blank. In other words, $\hat{\mathcal{D}}_i$ is the union of all knowledge states for which there exists a strategy of i or fewer steps guaranteed to attain the goal. [Note that we have $\mathcal{I}_0 \subseteq \hat{\mathcal{D}}_k$.] If ever the actual knowledge state K is a subset of the set $\hat{\mathcal{D}}_i$, then it is possible to guess between a collection of knowledge states from which goal attainment is possible. The guess involves at most n choices. If it involves exactly one choice, then the strategy is in fact guaranteed to attain the goal. In general, one must worry about false goal recognition, using now the knowledge state K in place of $\mathcal{I}_0$ in corollary 3.15. An applicability condition can now be stated, which simply says that for all possible execution traces the system always winds up in one of the $\{\hat{\mathcal{D}}_i\}$. In other words, no execution trace should ever enter a blank entry in the dynamic programming table. This is quite a difficult condition to state generally in any meaningful way, partly because one must now look at execution traces that may be longer than k steps, and partly because the false goal recognition condition enters into the picture. Instead we will state a weaker condition, then show how to satisfy it with a very simple assumption.

Definition. Recall that a randomized strategy repeatedly guesses its initial starting region K_i, then executes some guaranteed strategy for attaining the goal from K_i. The execution terminates either with goal attainment or failure. We will refer to each such guess and strategy execution as a *single guessing loop* of the randomized strategy.

Definition. We will say that a randomized strategy may be *reliably restarted* if, whenever it fails to attain the goal recognizably on a single guessing loop, it recognizably lies within its initial starting region $\mathcal{I}_0$.

The following claim establishes a nice complement to corollary 3.15. To verify that the strategy may be reliably restarted in general one of course needs to check the condition of the claim for all possible knowledge states K_i that the randomized strategy might guess (recall there are at most n of them). The claim is essentially a restatement of the definition of reliable restart, but with a slightly stronger condition.

Claim 3.16 *Assume the hypotheses of claim 3.14, and suppose that the guaranteed strategy for K_0 is assured of reliable goal recognition from K_0. The randomized strategy*

may be reliably restarted if and only if $\mathcal{I}_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k \subseteq \mathcal{I}_0$ *for all possible execution traces that might occur which fail to attain the goal recognizably and for which* $K_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ *is either empty or contains non-goal points.*

3.9.6 Observations and Assumptions

Notice that if a strategy both is assured of reliable goal recognition and may be reliably restarted for all relevant knowledge states K_i that cover $\mathcal{I}_0$, then whenever a single guessing loop of the randomized strategy is executed from the region $\mathcal{I}_0$, it is guaranteed to attain recognizably either the goal or again the region $\mathcal{I}_0$ itself. This condition is in appearance very similar to Donald's EDR condition (see page 100 of [Don89]), which insists that a strategy be guaranteed to attain recognizably either the goal or a region, called the *failure region*, from which success is not possible. One difference is that our failure region is the start region itself.

Another related difference is that the condition does not work in reverse. In other words, the converse statement that recognizable attainment of the goal or the start region implies reliable goal recognition and reliable restart is simply not true. After all, if the start region is the entire state space, then *any* strategy is guaranteed to attain recognizably either the goal or the start region, but the strategy need not satisfy the condition of reliable goal recognition.

The failure of the converse statement suggests that verifying reliable goal recognition and reliable restart are in general quite difficult. However, they are easily satisfiable conditions if we make two special assumptions.

Assumption of Goal Recognizability. First, we will assume that the goal is recognizable independent of any particular execution. This means that if the sensor signals goal attainment then the goal has indeed been attained, and conversely, if the goal is entered then the sensor will signal goal attainment.

Assumption of Covering Start Region. Second, we will assume that the start region for any guessing strategy is the entire state space. In general, one can relax this assumption by considering only that portion of the state space that might ever be traversed.

One final comment is in order. When the guessing strategy fails and decides to guess anew, it need in general not guess between the q [illegible] possible knowledge states that cover the starting region $\mathcal{I}_0$, but only between those knowledge states that cover the new start region $\mathcal{I}_0' = \mathcal{I}_0; A_1; I_1; A_2; I_2; \cdots; A_k; I_k$ determined by the most recent execution trace. This can sometimes speed up convergence. In particular, if $\mathcal{I}_0'$ is actually equal to one of the knowledge states for which a guaranteed strategy exists, then the randomized strategy is assured of convergence on the next attempt.

3.10 Comparison of Randomized and Guaranteed Strategies

Suppose one is in the fortunate situation of having both a guaranteed strategy for attaining a goal and a randomized strategy of the type just discussed. One question is whether it ever makes sense to use the randomized strategy. The answer is yes, assuming that the expected convergence time for the randomized strategy is significantly less than the convergence time for the guaranteed strategy. In order to set up this comparison, let us suppose that the guaranteed strategy for the starting state $\mathcal{I}_0$ is found in the ℓ^{th} column of the dynamic programming table, and let us suppose that the guessing strategy is found in the k^{th} column. Assume that there are q knowledge states $K_1, \ldots, K_q$ between which the randomized strategy guesses, and suppose that the guaranteed strategies for these states converge in steps $k_1, \ldots, k_q$, respectively. In other words, the worst-case convergence time for the guaranteed strategy for $\mathcal{I}_0$ requires ℓ steps, and the worst-case convergence time for K_i requires k_i steps $(i = 1, \ldots, q)$.

If we assume that the randomized strategy always guesses between all possible q states, then the expected time until convergence is bounded by $\sum_{i=0}^{q} k_i$, which in turn is bounded by $q\,k$. It is a little strange mixing these expected and worst-case times, but the idea is similar to the example involving random key selection in the introduction. Essentially, if $q\,k$ is on the order of ℓ, or larger, then it doesn't make much sense to use the randomized strategy. However, if $q\,k$ is considerably less than ℓ then it is probably a good idea to use the randomized strategy. In particular, if ℓ is exponentially large in the problem specification, and k is only polynomially large, then it always makes sense to use the randomized strategy. This is because, as we noted early in the chapter, the probability that the randomized strategy will require more than t attempts is less than $\left(\frac{q-1}{q}\right)^t$. Recall also that q is bounded by n. It follows that for fixed n, the strategy converges exponentially fast in the number of steps. One may worry that as n gets large q may also get large, in which case, $(q-1)/q$ approaches unity. This seems to imply that as n gets large one cannot guarantee fast convergence. Notice, however, that if $t > m\,q$, where m is some integer and q is large, then the probability of the randomized strategy requiring more than t steps is less than e^{-m}, so convergence is still fast. In particular, in quadratic time the probability of failure can be made exponentially small.

As an aside, consider how randomization by guessing relates to the labelling scheme discussed earlier (see section 3.5). Essentially all non-goal states are assigned the same label, namely the number k, while goal states are assigned the label zero. Then the expected velocity at all non-goal states is at least $-1/q$, when averaged over each step of a k-step strategy, and thus the expected convergence time is bounded by $k\,q$. In some sense, by considering composite steps consisting of k basic steps, we have transformed a non-deterministic problem into a two-state probabilistic problem.

3.11 Multi-Guess Randomization

Thus far we have only dealt with randomization by guessing the starting state of the system. In general, it is equally possible to consider sequences of several guesses. In other words, when executing a strategy, at some point a knowledge state is encountered that is the union of several smaller knowledge states. Instead of executing a strategy applicable to the larger knowledge state, a system could simply guess between the smaller states, then use strategies appropriate for each of these. In terms of planning, the standard preimage or dynamic programming approaches continue to apply, but with an additional operator. Call this operator SELECT. SELECT operates as follows.

An Augmented Dynamic Programming Table

First, let us augment the dynamic programming table. Each column in the dynamic programming table will contain three types of entries, namely BLANK, GUARANTEED, and RANDOMIZED. The intuition is that BLANK and GUARANTEED are as before. Specifically if the entry for a knowledge state K is a GUARANTEED entry then there exists a tree of actions that is guaranteed to attain the goal assuming that the initial state was indeed inside K. A BLANK entry implies, as before, that there is no such strategy, and, now, also that there is no strategy involving random choices. The RANDOMIZED label in the entry for a knowledge state K means that there is a tree of operations that has some probability of attaining the goal. The operations involve both standard non-deterministic actions and the guessing operator SELECT. It is sometimes also useful to distinguish between different RANDOMIZED entries based on the probability of success of attaining the goal by a particular sequence of guessing operations. For a given knowledge state, this number is easily computed as the minimum product of guessing probabilities along possible paths from that knowledge state to the goal. The probability represents the worst-case probability of attaining the goal by a sequence of actions and guessing operations. It does not take into account goal attainment that is possible even when a guess is wrong. For this reason the probability may considerably underestimate the actual probability of success, and places into question its utility. Nonetheless, in some situations these probabilities provide a useful lower bound for comparing different strategies.

Planning

And now for the augmented planning process. Suppose that the planner has backchained to the k^{th} column of the dynamic programming table, and is currently considering the $k+1^{st}$ column. First the planner fills in all entries using only the standard non-deterministic actions. In other words, for each knowledge state K, if there is an action $\hat{A}$ of the form $\hat{A}: K \mapsto K_1, \ldots, K_2$, and each of the K_i has a non-BLANK entry in the k^{th} column, then the entry for K in the $k+1^{st}$ column may be taken to be $\hat{A}$. If there are several such actions $\hat{A}$, then one may wish to distinguish between

different actions by considering the labels of the entries for the knowledge states K_i to which the action can transit. In particular, suppose RANDOMIZED entries actually have probabilities of success associated with them. Then it makes sense to assign the probability 0 to any BLANK entry, and the probability 1 to any GUARANTEED entry. One can then associate with each action $\hat{A}$ a worst-case probability of success (but recall that this may be an underestimate). Specifically, if p_i is the probability of success associated with the knowledge entry for K_i in the k^{th} column, then the probability of success $p_{\hat{A}}$ for $\hat{A}$ may be taken as $\min_i\{p_i\}$. If several actions $\hat{A}$ are applicable at the current knowledge state, one can then select that action which maximizes $p_{\hat{A}}$. In particular, if there is an action that only transits to GUARANTEED states, then the planner should select it. Similarly, if all actions have worst-case probability zero of success, then the planner should simply leave the entry for K BLANK. Once an action has been selected, it provides a label and/or a probability of success for the current knowledge state K.

Once the entries in the $k+1^{st}$ column have been filled in in this way, the planner next considers all remaining BLANK entries in that column. In particular suppose K is a knowledge state whose entry is BLANK. If the knowledge state can be written as a finite union of non-BLANK states $\{K_1, \ldots, K_q\}$, then the SELECT operator comes into play. It provides a transition from K to one of the K_i via randomization. The entry for K in the $k+1^{st}$ column becomes a RANDOMIZED entry, with worst-case probability of success given by $\frac{1}{q}\min_i\{p_i\}$, where p_i *is the worst-case probability of* success for state K_i in the $k+1^{st}$ column. Again, the planner may wish to use SELECT to point from K to a collection $\{K_i\}$ of minimal size, or perhaps to a collection that maximizes the worst-case probability of success.

As usual, one must ensure that reliable goal recognition and reliable restart are possible.

Execution

At run-time, suppose nominally there are k steps remaining and the current knowledge state is K. If the entry for K is BLANK, then execution of this particular guessing loop stops, and a new loop at the beginning of the table is restarted, if possible. If the entry for K is not BLANK, but contains an action $\hat{A}$, then the system executes that action, thereby proceeding to the $k-1^{st}$ column. If the entry for K is RANDOMIZED and thus contains a SELECT operation, then the system randomly chooses one of the $\{K_i\}$ specified by this SELECT operation, whereupon the action stored in the entry for the selected K_i is executed. If ever the goal is attained, execution stops. Starting or restarting the guessing loop entails determining an initial knowledge state by performing a sensory operation and intersecting the resulting sensory interpretation set with the set $\mathcal{I}_0$, in which all motions are assumed to occur. An alternative is to restart the guessing loop by considering the set $\mathcal{I}_0^{i+1} = \mathcal{I}_0^i; A_1; I_1; A_2; I_2; \cdots; A_k; I_k \subseteq \mathcal{I}_0$ in place of $\mathcal{I}_0$, where $\mathcal{I}_0^i$ is the initial knowledge state at the start of the i^{th} iteration of the guessing loop. This procedure preserves full history independent of any guesses, and thereby may limit the number of states between which the strategy must guess

on each new iteration.

Examples

For an example of multi-level guessing in the continuous domain, see the example of figure 4.8 on page 219. For a simpler example consider again the discrete approximation to the peg-in-hole problem of figure 2.13 on page 85. Suppose that the peg does not just fall into the hole once it is above the hole. Instead, the system first must ascertain that the peg is above the hole, then try to push down. If sensing is poor so that the system cannot decide on which side of the hole the peg is located, then the system may have to resort to a multi-level guessing strategy. In particular, the system first guesses on which side of the hole the peg is located, then moves in the goal direction specified by this guess. Next, the system repeatedly guesses whether it has moved the peg above the hole, and either pushes downward if it guesses "yes", or continues its motion if it guesses "no". If the system guesses correctly each time, then the peg will enter the hole. Let us assume that this success is recognized by some other means (for example, by considering the height of the peg above the hole). One could imagine removing the second set of guesses in this strategy, and instead always pushing down after each move. If this is feasible it will be generated as a strategy by the dynamic programming approach. However, perhaps pushing down disturbs some other parameter of the system whenever the peg is not above the hole. For instance, if the peg is gripped by a robot hand, the fingers might slide, and the peg might have to be regrasped from some initial configuration. In this case it might be better not to push down after each attempt. Another possibility is that there are multiple holes, so that pushing the peg down into the wrong hole requires extracting it again. In any event, both types of strategies may be generated by the dynamic programming approach.

Randomization Can Solve Nearly Any Task

Once one has an operator such as SELECT, one can solve any task for which there is some chance of attaining the goal! As usual, this assumes goal recognizability and reliable restart. In order to see that any problem is solvable, first recall claim 3.4. This claim tells us that whenever it is "certainly possible" to move from any state to the goal, then there actually exists a guaranteed strategy for attaining the goal, assuming a perfect-sensing function. Furthermore, this strategy requires at most $r = |\mathcal{S}| - |\mathcal{G}|$ steps. A guessing strategy may thus be constructed. The strategy simulates the perfect sensor by guessing the actual state of the system at each step of the perfect-sensing strategy, before deciding on the next action to execute. Of course, the worst-case expected execution time of such a randomized strategy may be quite bad. In particular, the probability of guessing the state correctly during all stages of an r-step strategy may be on the order of $1/r!$. Thus the worst-case expected execution time is $O(r\,r!)$.

3.12 Comments and Extensions

3.12.1 Randomization in Probabilistic Settings

The knowledge states in the probabilistic setting are probability distributions on the underlying state space. In other words, each knowledge state is an ordered n-tuple of non-negative numbers that add up to one, where $n = |\mathcal{S}|$.

If, as we have assumed, all of the underlying states are connected to the goal, then for each state one can determine a sequence of at most r transitions leading from the state to the goal. Here r is the number of non-goal states, as usual. The probability of actually executing this sequence is at least equal to the product of the probabilities along each of the arcs. For each state one can easily determine (using Dijkstra's algorithm) a sequence of transitions of maximum probability. A randomized strategy of the flavor discussed for the non-deterministic case would consist of guessing the underlying start state of the system, then executing a sequence of actions corresponding to the sequence of transitions thus determined. The probability of attaining the goal is then at least equal to the probability of guessing the correct start state, multiplied by the probability of actually executing the sequence leading from that state to the goal. This probability is bounded from below by $\frac{1}{r}p^r$, where p is the smallest probability appearing on any arc in the r sequences of transitions leading to the goal. This number may be quite small in general. Of course, if there exists a guaranteed strategy for attaining the goal, assuming perfect sensing, then there exists a guessing strategy just as for the non-deterministic case above. For both types of randomized strategies, it is assumed that the goal is reliably recognizable.

In general, however, if one has probabilities available for the actions and sensors, then it does not make much sense to randomize in the way one might do for the non-deterministic case. In particular, the probability of executing a sequence of transitions from a state to the goal is often a severe underestimate of the actual probability of attaining the goal. This was made clear by the examples on random walks. Instead of constructing strategies that randomize by guessing, it is generally more useful either to construct strategies that make local progress or to solve the complete Markov Decision Problem and try to minimize the expected time to attain the goal.

There is one special form of randomization that does appear fairly directly in the probabilistic setting. This consists of moving the state of the system in order to change the probability distribution over the state space, say to equalize it. This randomization is useful for some tasks where it is desired to meet some action's preconditions at least probabilistically. The main purpose of this randomization in the domain of manipulation is to blur environmental details. A natural setting is in tasks that involve geometric uncertainty. An example is given by a peg-in-hole problem in which the location of the hole is not modelled accurately. By randomizing the peg's position near the hole, a robot can in many cases ensure that there is a non-zero probability of starting from a location from which goal attainment is possible.

The parts-sieving example of chapter 1 tried to make a similar point. In that example the geometric uncertainty was in the exact shape and size of the sieve

elements.

In general, given an action (or composite action consisting of several actions), that is to be repeated over and over, one can determine the steady state distribution over the state space using the theory of Markov chains as discussed in section 3.4.1. One can compare different actions in terms of the final distribution attained, and in terms of the expected time until steady state is achieved.

In some cases, the actions required to attain a particular randomization may be clear from context. For instance, in order to achieve a uniform distribution over a bounded one-dimensional lattice, it suffices to perform a standard one-dimensional random walk, with reflection at either ends of the lattice. There has been considerable work on estimating the time required for convergence to a uniform distribution for random walks on lattices (see for instance the article on card-shuffling [AD]). Related work dealing with random walks on graphs includes [GJ], [AKLLR], [SJ], [CRRST], and [Z].

3.12.2 Randomization: State-Guessing versus State-Distribution

The previous sections have indicated how a system can probabilistically attain a goal by randomly choosing between several guaranteed strategies, whose applicability conditions individually cannot be met, but which are met when taken as a disjunctive collection. This form of randomization has a different flavor than the randomization indicated in the early sections of the chapter, namely in the gear-meshing and parts-sieving examples (see also section 1.2). In those tasks, there was a single action that would attain the goal, given that the action's pre-conditions were met. The pre-conditions could not be satisfied with certainty, but could be satisfied probabilistically by randomly moving the system about, such as by twirling the gears or shaking the sieve. The randomization in these cases seems more direct, since it actually randomizes the state of the system, than does the randomization achieved via guessing. However, these two forms of randomization are actually very similar. In particular, suppose that some knowledge state K is a precondition to action A, where action A is guaranteed to achieve the goal $\mathcal{G}$. Now suppose that the initial state of the system is known only to lie in some set $\mathcal{I}_0$ that contains K. The state-distribution approach consists of randomizing the states within $\mathcal{I}_0$, so that there is some non-zero probability of actually being in the set K. [If it is true equalization, then that probability is $|K|/|\mathcal{I}_0|$.] This means in particular that it is "certainly possible" to reach K from any state in $\mathcal{I}_0 - K$. Thus there must be a perfect-sensing strategy for attaining K, and hence a randomization by guessing strategy for attaining $\mathcal{G}$, from any point in $\mathcal{I}_0$. [As usual, it is assumed that the goal is recognizable reliably and that the guessing strategy may be restarted reliably.] Conversely, suppose that there exists a guessing strategy for attaining K. Then in some sense there exists a strategy that randomizes the state of the system. After all, if one considers all possible guesses in the guessing strategy, these define a random collection of action sequences that randomize the state of the system. However, it need not be the case that there is a

well-defined distribution over $\mathcal{I}_0$, nor that all states of $\mathcal{I}_0$ are necessarily reachable.

More generally, the set K may not be known, of course, which is why true randomization via state-motion may be required. Formally, however, this presents no problem in drawing a connection between randomization via state-distribution and randomization via state-guessing. This is because one can often augment the underlying state space with an extra dimension that encodes the parameters whose unknown values define K. See [Don89] for further details on handling model uncertainty. In other words, one may not know whether it is possible to get from some state to the goal under some action, so sometimes one guesses that it is possible and executes the action, whereas at other times one guesses that it is not possible, and instead moves to a completely different state.

3.12.3 Feedback Randomization

In the previous guessing strategies extensive use was made of history. Certainly history plays a major role within each of the guaranteed strategies. Indeed, new knowledge states are formed from old ones by forward projecting the effect of actions, then intersecting these with sensory interpretation sets. Similarly, each time the guessing strategy randomly selects a particular knowledge state, it is effectively assuming a particular history. All actions following this random selection update knowledge states in the usual manner, so that the derived history is correct in so far as the random selection was correct.

The process of guessing history can be extremely useful when a strategy depends on extensive history to prune possible sensory interpretations. If sensing uncertainty is large, it might otherwise never be possible to select the correct motions to perform. By guessing some of this history, goal attainment is possible, at least, if the guess is correct. On the other hand, in some cases, if the guess is incorrect, it may take several steps of execution before an inconsistency is detected or before failure to attain the goal terminates the loop. In particular, in the case of no sensing (except for goal recognition), a guaranteed strategy that has been randomly selected may have to run its full course before the system can recognize goal failure. For instance, imagine that one has the diagram for a maze in a cave, but is blindfolded (and not allowed to purposefully feel one's way along the walls of the cave). So sensing is very limited. Suppose, however that one can turn fairly accurately and can measure distance by walking fairly accurately, so that one can actually follow the map well, based purely on dead reckoning. In other words, control and thus history are very good. Thus, if one knows one's starting position or can guess it fairly accurately, then one has a good chance of getting out of the cave quickly, whereas if one can only guess one's starting location with enormous uncertainty, then the time required may be proportional to the size of the cave times the time required to execute a single attempt to exit the cave.

Using Current Sensed Information Only

An alternative to retaining history in updating the knowledge state after each motion is to simply use the state of knowledge returned by the current sensory value. More generally, the constraints imposed by one's hardware or timing considerations may require that one design strategies whose actions are based solely on current sensed values, and not on history. For this reason it is natural to consider approaches for synthesizing simple feedback loops.[4] Consider the representation of actions. Suppose that the effect of an action A on knowledge state K_1 is K_2, and that the range of possible sensory interpretation sets associated with K_2 is $\Xi(K_2) = \bigcup_{s \in K_2} \Xi(s) = \{I_1, I_2, \cdots, I_\ell\}$. In the framework developed thus far, one models the induced action $\hat{A}$ as

$$\hat{A}: K_1 \mapsto K_2^1, K_2^2, \ldots, K_2^\ell,$$

where $K_2^i = K_2 \bigcap I_i$. In a framework without history one models the action simply as

$$\hat{A}: K_1 \mapsto I_1, I_2, \ldots, I_\ell.$$

The first expression models history, the second only models possible sensing information. Thus the only knowledge states that are relevant are those corresponding to possible sensory interpretation sets.

Clearly, fewer tasks are solvable in a guaranteed sense with this type of approach, since it is in general more difficult to constrain the apparent state of the system. From a probabilistic point of view, solvability has not changed. This is because the existence of a randomized strategy depends only on goal reachability, a condition that may be checked by determining whether a perfect-sensing strategy exists. For a perfect sensor, history adds no extra information. Of course, once one tries to simulate the perfect-sensing strategy using an actual sensor and a guessing strategy, the quality of one's knowledge states determines the expected time until the goal is attained. For a purely sensor-based system, that is, a system without history, although all tasks are still solvable probabilistically, the expected convergence time will in general increase.

As an example consider a simple peg-in-hole problem. Either the two-dimensional peg-in-hole of figure 2.2 or the abstraction of the three-dimensional peg-in-hole discussed in section 1.1 are possible examples. A perfect-sensing strategy might consist of moving straight towards the hole. However, if there is sensing uncertainty and the system does not retain history, then it will become confused near the hole. Instead of relying on accurate information, the system effectively must guess where it is located. This manifests itself in the execution of a random action. The difference between history-based and simple feedback loops is particularly striking in the example of figure 2.2. In this example the motions are one-dimensional. Thus a randomized strategy that retained history could simply make a single guess,

[4] *Simple feedback* refers to the feedback of current sensed values without retaining past sensed values.

executing a long motion to attain the goal. Should failure occur, the strategy would then possess enough information to direct it towards the goal accurately. However, a simple feedback loop that does not retain history would make repeated guesses, effectively executing a random walk on one of the edges near the hole until it attained the goal. Thus in this example the difference between retaining history and only considering current sensory information manifests itself as the difference between linear and quadratic expected convergence times.

Using the full sensory interpretation set at each step rather than intersecting it with past history has at least one desirable characteristic, namely it preserves truth. In contrast, a guessing strategy that assumes a particular history need not preserve truth. Indeed the truth is fudged in order to provide a minimum probability of success. However, in some cases, namely those in which an adversary cannot force indefinite failure, and in which progress towards the goal is possible on average, a feedback loop based on current sensed values can provide reasonable convergence times while preserving accurate knowledge at each step of the strategy.

Progress Measures

One nice property of a perfect-sensing plan is that it places an implicit progress measure on the underlying state space. This was made explicit by claim 3.12. Such a simple progress measure on the underlying state space is not as easily provided by plans that involve general knowledge states, simply because a state may be a member of several different knowledge states that have different labellings. Only in the perfect-sensing case is there necessarily a unique labelling of states. This labelling plays the same role that the duration labels did in the Markov chain case. We observed earlier that the Markov chain model applied even in the imperfect-sensing case, so long as the action taken at any time was solely a probabilistic function of the state of the system (in particular, time and history invariant). A similar statement applies in the non-deterministic case, so that it makes sense to think about progress measures even with imperfect sensors. We alluded to this in section 3.6, but now is a good time to take a closer look. The discussion should tie together the concepts of progress measures and randomization by guessing in the setting of strategies that rely purely on current sensory feedback and not on history.

Feedback with Progress Measures

Suppose the collection $\{S_j\}_{j=0}^{\ell}$ is given as per claim 3.12 for some discrete planning problem with non-deterministic actions. Let the label for each state s simply be the index j of the unique set S_j that contains the state s. Define, as in section 3.6, the worst-case velocity $v_{A,s}$ relative to some action A at some state s to be the maximum possible change in labellings, where the sign of the change is significant. An action is said to make progress at a state s precisely when $v_{A,s}$ is negative.

Now consider how a simple feedback strategy operates. At any instant it has available some sensory interpretation set I. Given this sensory information the

strategy executes some action A. We are assuming that the choice of action A depends only on the sensory information and not on any hidden state variables that encode history or the passage of time. Thus A is either uniquely determined by I or chosen randomly from a collection of actions that is uniquely determined by I.

If one wishes to ensure that at each step the strategy makes progress relative to some labelling, then it must be the case that all non-goal states $s \in I$ transit to a state with lower label when A is executed, and all goal states remain in the goal. This in turn implies that there is actually a guaranteed strategy for attaining the goal, assuming that the goal is reliably recognizable once entered. Furthermore, the strategy converges in no more than ℓ steps, where ℓ is the highest possible label assigned to a state. The guaranteed strategy operates simply by executing that action A that ensures progress for all states in I, whenever the sensory interpretation set is I.

Planning Limitations

Before we comment on the generality of this approach, let us observe that even though there exists a guaranteed strategy whenever progress is ensured for all possible sensory interpretation sets, it need not be the case that a planner that only considers knowledge states corresponding to sensory interpretation sets can actually construct this strategy. This is because some notion of history is required in order to recognize convergence of the strategy, even though the strategy itself does not make use of history. In particular, the planner may not be able to synthesize the relevant progress measure.

For a simple example, consider figure 3.18. There are four states and two actions. Action A_1 is guaranteed to move state s_1 to the goal s_G, while it moves state s_3 non-deterministically either to state s_1 or state s_2. It leaves all other states unchanged. Similarly, action A_2 is guaranteed to move state s_2 to the goal, while non-deterministically moving state s_3 to either s_1 or s_2. Suppose that the set of sensor values is given by three possible interpretation sets, namely $\{s_1, s_3\}$, $\{s_2, s_3\}$, and $\{s_G\}$. So goal recognizability is ensured.

This example might be an abstract version of the two-dimensional peg-in-hole problem of figure 2.2, with an additional state corresponding to the placement of the peg in free space. The analogous sensing would be to assume that the system can distinguish on which side of the hole the peg is located, but that the system cannot decide whether the peg has made contact with a surrounding edge, as opposed to being in free-space above the hole.

A guaranteed simple feedback strategy for attaining the goal is of the form:

- If the sensory interpretation set is $\{s_1, s_3\}$, then execute action A_1.
- If the sensory interpretation set is $\{s_2, s_3\}$, then execute action A_2.
- If the sensory interpretation set is $\{s_G\}$, then terminate successfully.

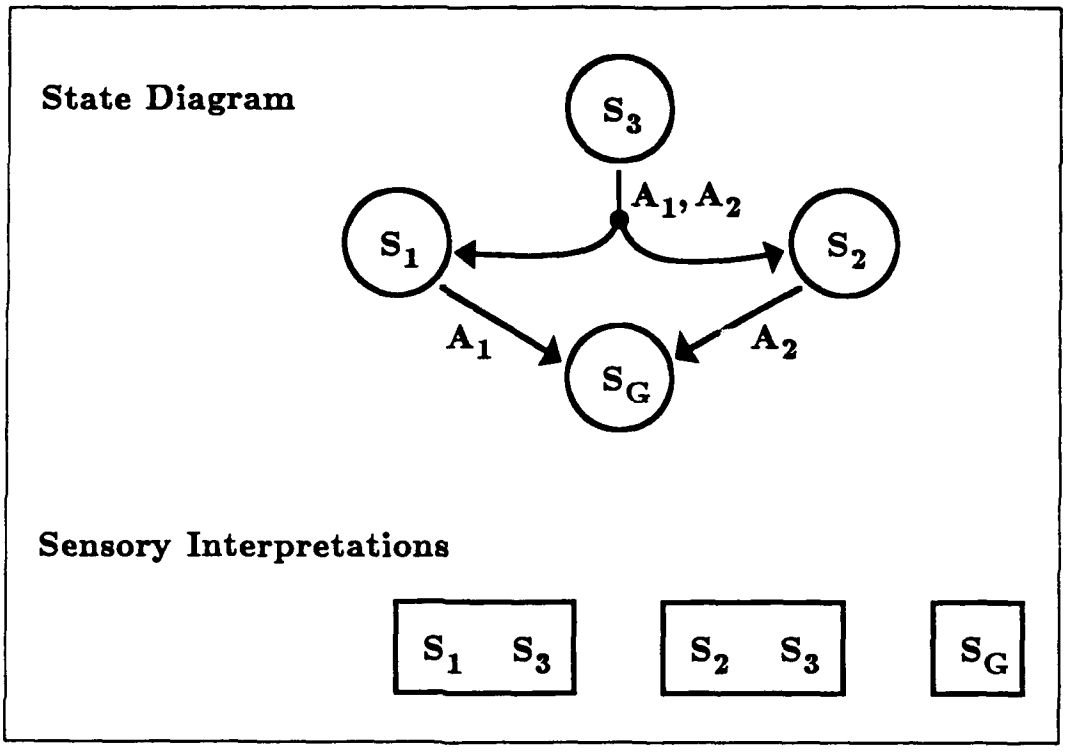


Figure 3.18: For this state diagram and the collection of possible sensory interpretation sets, there exists a guaranteed strategy for attaining the goal. Furthermore, the strategy does not require history to execute. However, a backchaining planner that ignores history cannot generate the strategy. [The sensory interpretation sets are indicated by rectangles surrounding the states.]

The strategy is guaranteed to succeed because at each step it ensures progress relative to a progress measure that labels s_G with 0, s_1 and s_2 with 1, and s_3 with 2. Observe, however, that if a planner only considered the three knowledge states given by the sensory information, then it could not backchain even one level. This is because, for example, there is no action that guarantees that the knowledge state $\{s_1, s_3\}$ is transformed into either of the other two knowledge states. Of course, a planner that made full use of history would be able to synthesize a guaranteed strategy for attaining the goal.

More generally, consider a strategy that only uses current sensory feedback at execution time, but is guaranteed to converge to a goal because it is assured of local progress relative to some labelling. Then there need not be a solution visible to a planner that only considers knowledge states that are possible sensory interpretation sets, but there always will be a solution visible to a planner that considers full history and sensing information (this, in the preimage setting, is a consequence of Mason's completeness result [Mas84]). After all, the history available to the execution system (and the planner) must be at least as constraining as the implied history of the progress measure. However, a planner that uses full history in synthesizing a guaranteed strategy need not find a strategy that is necessarily executable using only a simple feedback system. This is because the planner may specify different actions for two knowledge states that can give rise to the same sensory interpretation set at run-time. Some additional mechanism would be required to ensure that a stationary strategy based purely on current sensory information is derivable from the guaranteed strategy suggested by the planner.

As an example, suppose in the previous figure there is a third action A_3 whose effect on state s_3 is to move non-deterministically to one of the states s_1 or s_2. All other states are left unaffected by this action. Then a possible backchaining table for a guaranteed strategy might be of the following form. [Notice that not all knowledge states are needed in determining a guaranteed plan. For instance, if we assume initial sensing, then the knowledge state $\{s_1, s_2, s_3\}$ is easily ruled out.]

Steps Remaining				
2	1	0		
$\hat{A}_3$			$\{s_1, s_3\}$	Knowledge States
$\hat{A}_3$			$\{s_2, s_3\}$	
$\hat{A}_1$	$\hat{A}_1$		$\{s_1\}$	
$\hat{A}_2$	$\hat{A}_2$		$\{s_2\}$	
stop	stop	stop	$\{s_G\}$	

Actions guaranteed to attain the goal.

Now observe, that at run-time, if the actual state of the system is s_1, then the sensor will return the interpretation set $\{s_1, s_3\}$. The table would say to execute A_3 and sense, but, of course, that is not the right thing to do in state s_1. Similarly for s_2. If by chance the planner had returned the same table, but with $\hat{A}_1$ and $\hat{A}_2$ in

the appropriate places instead of $\hat{A}_3$, then a consistent stationary simple feedback strategy would have been obtainable. The problem is that just running the planner does not ensure such a policy.

Progress as a Generalization of Guarded Moves

This discussion indicates that progress measures form a useful intermediate planning approach, situated between strategies that employ perfect sensing and those that rely on full history. In many cases the progress measure is naturally derived from the perfect-sensing strategy, although arbitrary progress measures are imaginable. The progress measure approach is a natural generalization of those strategies that execute a single action over and over until some sensory condition is met (see the discussion on guarded moves on page 46). For instance, the underlying primitive of the preimage methodology is a single command that is executed until some termination condition is true (see chapter 4). Moving down until one feels a force of collision is a typical application of such a primitive action. In the discrete context this primitive corresponds to moving through a progression of states under the repeated application of a single action, until some goal is attained. The progress measure is simply the distance moved, or perhaps the change in some coordinate. The notion of progress is of course more general than progress relative to a single action, and much of this chapter has been concerned with generalizing that notion. The more general notion involves categorizing states by how far they are from the goal in terms of how many actions may be required maximally to attain the goal, as discussed in claim 3.12.

Guessing, Whenever Progress is not Possible

Unfortunately, it may not always be possible to ensure that progress is made at every state for every possible sensory interpretation set that might arise while the system is in that state. In these cases it is useful to randomize by guessing as before. In other words, if some sensory interpretation set is of the form $I = \bigcup_{i=1}^{q} K_i$, such that there are actions A_i that cause every state in K_i to make progress, then the system should randomly choose one of the A_i to execute. This guessing is similar to the guessing employed in the randomization of section 3.9. The difference is that now the knowledge state of the system is the most recent sensory interpretation set, rather than a state derived from previous guesses and actions. One imagines that in the worst case each step of the strategy requires an n-way guess. Such could be the case in a sensorless task (sensorless except for goal recognizability). However, in that setting one would probably do well to employ some form of history.

Sensing and the Speed of Progress

Let us discuss the role of sensors in determining whether progress is possible at a given state. Consider a state s and its collection of possible sensory interpretation sets $\Xi(s)$. If for all sensory interpretation sets $I \in \Xi(s)$, it is possible to select an action A_I that ensures progress independent of the actual state $\hat{s} \in I$, then in particular it is possible

to ensure progress at s. Furthermore, if one considers the sensor to be adversarial, then one may assume that the sensor always forces that interpretation set I for which the action A_I makes the least amount of progress at state s. Thus it makes sense to define the worst-case velocity at s to be

$$v_s = \max_{I \in \Xi(s)} v_{A_I,s},$$

which agrees with the definition (3.20).

By similar reasoning, if the sensor is adversarial, and there is some possible interpretation set $I \in \Xi(s)$ for which progress is not ensurable independent of the actual state giving rise to I, then progress may not be guaranteed at s. Instead the action to be executed is chosen probabilistically from some collection $\mathcal{A}_I = \{A_1, \cdots, A_q\}$ that corresponds to the collection of knowledge states $\{K_i\}$ that cover I. In this case, it makes sense to define a worst-case average velocity, namely as:

$$v_s = \max_{I \in \Xi(s)} \left\{ \frac{1}{|\mathcal{A}_I|} \sum_{A \in \mathcal{A}_I} v_{A,s} \right\}.$$

The point is that whenever the system is in state s and sensory interpretation set I occurs, on average the guessing strategy will make progress that is at least $-(\sum_{A \in \mathcal{A}_I} v_{A,s})/|\mathcal{A}_I|$. Thus an adversarial sensor can only try to minimize this quantity by selecting sensory interpretation sets I that behave poorly. Once, again, if v_s is negative for all states and bounded away from zero by v, then the worst-case average execution time will be bounded by the maximum label divided by $-v$.

This process generalizes as one changes adversarial actions to probabilistic actions, and/or adversarial sensors to probabilistic sensors, until one eventually gets a process resembling the Markov chains discussed earlier in this chapter.

3.12.4 Partial Adversaries

In the discussion on non-deterministic tasks thus far, it has been assumed that an adversary can always force the worst possible motion or sensing information at any instant at any state. However, for some physical tasks the non-determinism specified in the actions and sensing function is due to a paucity of knowledge in modelling the system, rather than the existence of an actual adversary. In other words, the actual transitions or sensor values obtained depend on some set of parameters whose exact values are unknown, and hence are modelled as non-deterministic uncertainty. The actual system behaves in a manner consistent with a particular instantiation of these parameters. This means that the range of transitions possible in response to an action and/or the sensory interpretation sets obtained from a sensor are coupled at different states of the system. In short, if an adversary can choose a bad transition at some state, this may reflect a particular instantiation of the unknown parameters that precludes an independent worst-case choice at some other state.

As an example, consider the case of a sensor with an unknown bias. Specifically, suppose that there is a sensor, that returns a sensed position x^* that lies within some

error ball about some unknown bias, denoted by $B_\epsilon(x + b)$. The notation is meant to convey the idea that the actual state is x, and that the error ball is centered at a point that is offset from x by some bias b, and has radius ϵ. This notation makes a lot of sense in a vector space such as $\Re^n$, in which the error ball might represent the support of some distribution function describing the possible sensor values (see also the examples of sections 2.2.2 and 2.4). Conceptually, we can imagine a similar situation for discrete tasks. If the bias b is known, then whenever one sees a sensor value x^*, the resulting sensory interpretation set implies that the actual state of the system must lie in the set $B_\epsilon(x^* - b)$. However, if the value of b is not known exactly, but can merely be bounded in magnitude, say as $|b| \leq b_{\max}$, then one can merely assert that the actual state of the system lies in the set $B_{\epsilon+b_{\max}}(x^*)$. One would model this non-deterministically by saying that the sensing function Ξ can return for each state x one of a collection of error balls of radius $\epsilon + b_{\max}$, namely the collection $\{B_{\epsilon+b_{\max}}(x^*)\}$, as x^* varies over $B_{\epsilon+b_{\max}}(x)$. This suggests that if the state of the system is x, then an adversary could choose any sensor value x^* that lies within the error ball $B_{\epsilon+b_{\max}}(x)$. Of course, that is not true. An adversary can merely choose any sensor value from the range $B_\epsilon(x + b)$, for some actual but unknown b.

Now consider a task in which the non-determinism is so great that there is no strategy that ensures progress at each state, relative to some labelling. However, suppose further that there exists an unmodelled parameter, such as the bias b in the previous example, whose instantiation would permit progress for a large number of states. In other words, given a particular instantiation of this parameter one can devise a strategy for which the mix of states at which progress is possible and states at which progress is not possible is sufficient to ensure goal attainment within some time bound. If this strategy is actually independent of the particular instantiation of the unknown parameter, then the strategy is assured of goal attainment within the desired time bound.

An example is given again by the sensing bias mentioned above. If the task is to move to some region based on sensor values, then for certain approach directions the bias will aid in attaining the goal, while for other approach directions the bias will hinder attainment (recall the peg-in-hole example of section 1.1). One can take advantage of the bias, without knowing its true value, simply by executing a strategy of the type discussed in this section. Specifically, whenever the system can make progress towards the goal, it does so, and otherwise it executes a random motion. The random motion ensures that if the system is in a region in which the bias is precluding sensory interpretation sets that ensure progress towards the goal, then eventually the system will either attain the goal or drift out of that region and into another region within which the bias facilitates goal attainment.

3.13 Some Complexity Results for Near-Sensorless Tasks

In this section we consider a special form of the discrete planning problem, namely one in which the sensors provide no information other than to signal goal attainment. We shall refer to such problems as *near-sensorless.* Sensorless tasks form an important subclass of the set of robot tasks. Mason (see, for example, [Mas85] and [Mas86]) has studied these problems extensively. The motivation for studying sensorless problems stems from the realization that almost all tasks involve some operations in which the mechanics of object interactions dominates any informational content provided by the sensors. For instance, in grasping or pushing objects, even if sensors are available to provide a general sense of the object's behavior, the behavior of the object at the instant of contact tends to lie below the resolution of the sensors. Thus it is important to understand the behavior of objects, and the manner by which one can control them, in the absence of sensory information. [Brost86] and [Pesh] have further explored sensorless grasping and pushing. [MW], [EM] and [Nat86] have looked at other tasks that are amenable to sensorless solutions, such as the problem of unambiguously orienting an object given complete uncertainty as to the object's initial configuration, and [Wang] has studied extensively the impact problem.

In terms of the previous discussion in this chapter, we have seen that tasks in which sensing is perfect can be solved very quickly. For fixed control uncertainty, one may thus view sensing uncertainty as the devil that confounds one's guaranteed strategies. Indeed, randomized strategies were formulated precisely as a means for pretending to reduce sensing uncertainty, by simply guessing the state of the system, that is, by guessing the correct sensor value. Thus it is natural to look at the extreme case in which there is no sensing whatsoever. However, in order to satisfy the goal recognition criterion, we will insist that the goal be recognizable. In short, there is some sensing, but it is limited to deciding whether or not the goal has been attained.

In this section we will first briefly outline how the general backchaining planners discussed earlier specialize to the sensorless case, then indicate that sensorless and near-sensorless tasks are essentially equivalent from the point of view of generating guaranteed strategies. The main thrust of this section, however, is given by three examples that indicate the complexity of planning with and without randomization. We know, of course, from [PT] that planning solutions to discrete tasks in the absence of sensing is NP-complete. Specifically, for probabilistic problems in which there are costs associated with transitions, the problem of deciding whether or not there is a sensorless solution of a fixed number of steps that incurs zero cost is NP-complete. The three examples in this section elaborate on this type of result. Specifically, we will look at non-deterministic problems, and merely ask for the existence of a solution, not the existence of an optimal solution. This is equivalent to assigning costs that are either zero or infinite, depending on whether the goal is attained or not. Furthermore, we are interested in the comparison between guaranteed solutions and randomized solutions.

All three examples are abstract examples on graphs. Whether these can be actually realized by physical devices is not investigated. However, at the end of this section we indicate a physical device that has some of the same properties as the first example. The first example demonstrates a task for which there exists a guaranteed strategy, but which requires exponential time to plan and execute. In addition, there exists a randomized strategy that only requires quadratic expected time to attain the goal. This example indicates that some problems can actually be solved more quickly by randomized strategies than by guaranteed strategies. The second example indicates that not all problems can be solved quickly by randomization. And the third example shows that the particular planning approach used may investigate an exponential number of knowledge states even when the number of plan steps is fixed.

3.13.1 Planning and Execution

Let us briefly outline how a system might plan solutions to tasks in the sensorless and near-sensorless settings. Towards this goal, it is useful to consider the effect that actions and sensing operations have on knowledge states. Recall the notation of section 3.9.

Suppose that a system is initially in knowledge state K and suppose that at execution time a sequence of actions $\{A_1, \ldots, A_k\}$ is executed yielding a sequence of sensory interpretation sets $\{I_1, \ldots, I_k\}$. The final knowledge state resulting from this particular execution trace is given by $K; A_1; I_1; \cdots; A_k; I_k$. In general, of course, a plan might specify a decision tree, so that the actions executed are themselves functions of the observed sensory information. In the sensorless case, the sensing at each stage provides no additional information, so one can write the execution trace as $K; A_1; \mathcal{S}; \cdots; A_k; \mathcal{S}$. In particular, it is possible to decide before execution whether or not the resulting knowledge state is inside the goal set $\mathcal{G}$.

This simplifies planning greatly. It means that all actions may be viewed as *deterministic* transitions in the space of knowledge states. Recall that in converting an action on the underlying state space into an action in the knowledge space, it was only the intersection with possible resulting sensory interpretation sets that introduced any non-determinism. (See page 143.) Backchaining using dynamic programming thus entails determining whether there is a path from K to $\mathcal{G}$ in the directed graph whose states are knowledge states and whose arcs are the possible deterministic transitions specified by the actions. [This was essentially the approach taken by [EM] and [MW] in planning sensorless orienting strategies.] In short, a guaranteed strategy consists of a linear sequence of actions, not a general decision tree.

In the near-sensorless case each of the sensory interpretation sets I_i is either the whole non-goal space $\hat{\mathcal{S}} = \mathcal{S} - \mathcal{G}$ or the goal set $\mathcal{G}$. If we assume that an execution trace stops once the goal is attained, then each successful execution trace is of the form $K; A_1; \hat{\mathcal{S}}; A_2; \hat{\mathcal{S}}; \cdots; A_{k-1}; \hat{\mathcal{S}}; A_k; \mathcal{G} \subseteq \mathcal{G}$. Clearly, in this case the actions are indeed functions of the sensory information. In particular, the number of actions executed depends on when the goal is entered, an event that is only determined in a non-deterministic fashion at execution time. However, as in the sensorless case, for a

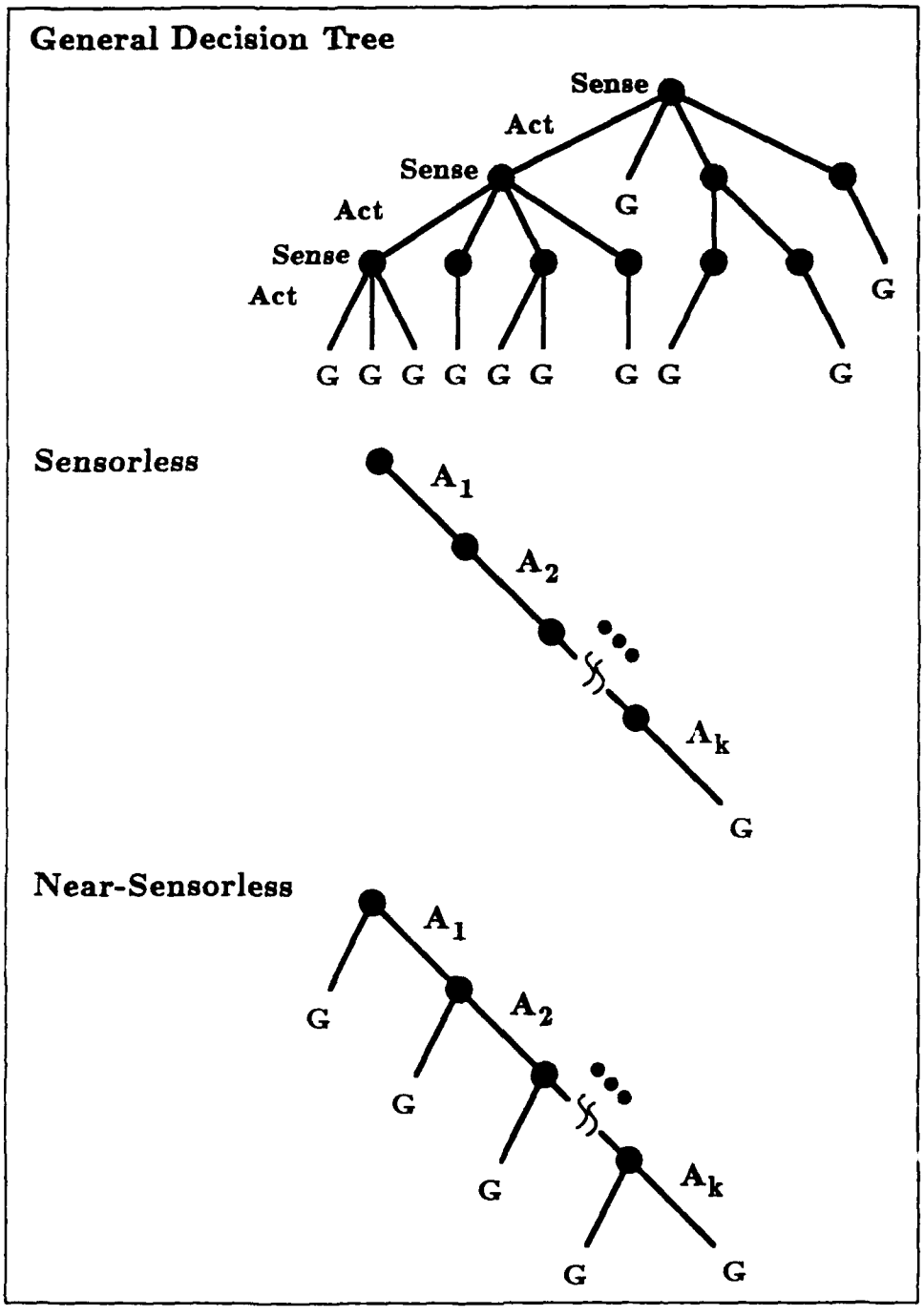


Figure 3.19: Decision trees for different types of strategies.

guaranteed strategy there is a definite sequence of actions that will be executed if the system does not enter the goal. Said differently, the decision tree is not really a general tree, but rather a linear sequence with one step branches at each step corresponding to early goal attainment. See figure 3.19.

In the space of knowledge states all actions are thus either deterministic or non-deterministic with two possible target states. In particular, suppose K is a knowledge state and A an action. If the forward projection $F_A(K)$ contains no goal states then one has a deterministic transition $A : K \mapsto F_A(K)$. Otherwise, one either has a non-deterministic transition $A : K \mapsto F_A(K) - \mathcal{G}, \mathcal{G}$; or complete goal attainment $A : K \mapsto \mathcal{G}$. Again, planning by backchaining corresponds to determining a path from K to $\mathcal{G}$ in a directed graph. As before, the states of the graph are the knowledge states. The arcs are simply the non-sensing transitions specified by the actions. This means that there is an arc labelled with A directed from K to $F_A(K) - \mathcal{G}$ whenever $F_A(K)$ contains at least one non-goal state, and otherwise there is an arc directed from K to $\mathcal{G}$ labelled with A. A sequence of such directed arcs leading from K to $\mathcal{G}$ represents the longest possible execution trace of the guaranteed strategy for attaining the goal.

One sees then that planning in the sensorless and near-sensorless settings are very similar. In the sensorless case one seeks a sequence of actions $\{A_1, A_2, \ldots, A_k\}$ such that $K; A_1; \mathcal{S}; A_2; \mathcal{S}; \cdots; A_k; \mathcal{S} \subseteq \mathcal{G}$, while in the near-sensorless case one seeks a sequence of actions such that $K; A_1; \hat{\mathcal{S}}; A_2; \hat{\mathcal{S}}; \cdots; A_k; \mathcal{S} \subseteq \mathcal{G}$. Here $\hat{\mathcal{S}} = \mathcal{S} - \mathcal{G}$ is the set of non-goal states. In the sensorless case the entire sequence of actions is always executed, while in the near-sensorless case the entire sequence is only executed in the worst case.

3.13.2 Partial Equivalence of Sensorless and Near-Sensorless Tasks

In the previous discussion, we saw a strong similarity between sensorless and near-sensorless tasks in terms of the structure of guaranteed solutions. The following paragraphs will make this similarity more precise.

Consider a discrete planning problem $(\mathcal{S}, \mathcal{A}, \Xi, \mathcal{G})$ in which the sensing function returns no information. In other words, $\Xi(s) = \{\mathcal{S}\}$ for every state s. Now consider a modified problem $(\mathcal{S}', \mathcal{A}', \Xi', \mathcal{G}')$, for which the set of states is augmented by one new state s_G, which now becomes the goal state. In other words $\mathcal{S}' = \mathcal{S} \bigcup \{s_G\}$ and $\mathcal{G}' = \{s_G\}$. Furthermore, define $\mathcal{A}'$ essentially to be just $\mathcal{A}$ with one additional action A_G, whose effect we will describe shortly. In particular, for any action $A \in \mathcal{A}$, let A have precisely the same effect on states in $\mathcal{S}$ as before, and let its effect on the new state s_G be a self-transition. In other words, $A : s_G \mapsto s_G$. The additional action A_G is designed to move any goal state in the old system into s_G, and otherwise non-deterministically move to any one of the states in $\mathcal{S}$. In other words, if the states of the original system are given by $\mathcal{S} = \{s_1, \cdots, s_n\}$, with goal states $\mathcal{G} = \{s_1, \cdots, s_r\}$, then A_G is specified by:

$$
\begin{array}{rcl}
A_G: & s_1 & \mapsto & s_G \\
 & \vdots & \vdots & \\
 & s_r & \mapsto & s_G \\
 & s_{r+1} & \mapsto & s_1, \ldots, s_n \\
 & \vdots & \vdots & \\
 & s_n & \mapsto & s_1, \ldots, s_n \\
 & s_G & \mapsto & s_G
\end{array}
$$

Finally, define a new sensing function Ξ' that gives partial sensing. In particular, Ξ permits goal recognizability in the new system. This is modelled as $\Xi'(s) = \{\mathcal{S}\}$ for every $s \in \mathcal{S}$, and $\Xi'(s_G) = \{\{s_G\}\}$.

Let us compare solutions to the two problems. Suppose that the unmodified system starts in knowledge state K and that there is a sequence of actions $A_1, \ldots, A_k$ and a sequence of (no-op) sensing operations such that at execution time the final knowledge state $K; A_1; I_1; \cdots; A_k; I_k$ lies inside the goal $\mathcal{G}$. Then clearly, for the modified system, the execution trace $K; A_1; I_1'; \cdots; A_k; I_k'; A_G; I_G'$ must be the singleton set $\{s_G\}$. Here each of the I_i' are the sensory operations returned by the modified sensing function Ξ'. Clearly $I_i' = I_i = \mathcal{S}$ for each $i = 1, \ldots, k$, and $I_G' = \{s_G\}$. Conversely, suppose that in the modified system there is a sequence of actions and sensing operations starting from some knowledge state $K \subseteq \mathcal{S}$, such that the final knowledge state is guaranteed to be the goal state $\{s_G\}$. Then, eliminating superfluous actions, clearly the last action must be A_G, and the execution trace up until this last action must be guaranteed to place the system into the original goal set $\mathcal{G}$, using only action in $\mathcal{A}$.

In short, if there is a strategy for knowingly achieving the goal in the sensorless system, then there is a strategy for knowingly achieving the goal in the near-sensorless system, and conversely. Thus the existence and structure of a guaranteed strategy for accomplishing a sensorless task is not fundamentally affected by the addition of a goal-sensor; one can always modify the problem slightly so that the goal-sensor does not provide any information useful to the guaranteed strategy. However, it is clearly true that in general, that is, for unmodified tasks, the goal-sensor does provide some additional information. In particular, if a motion happens to stray into the goal region, a goal-sensor will detect this. In contrast, a sensorless system would not necessarily be able to guarantee goal recognition. This property will be useful in the context of random strategies, as we shall see presently.

One can also establish a correspondence in the other direction, that is, one can convert any near-sensorless problem into a sensorless one with minor modifications, while preserving the existence and essentially the structure of guaranteed strategies. The basic idea is to replace the goal-sensor with a mechanical trap that precludes ever leaving the goal once it has been attained. So, suppose we are given a discrete planning problem $(\mathcal{S}, \mathcal{A}, \Xi, \mathcal{G})$ in which the state space is $\mathcal{S} = \{s_1, \cdots, s_n\}$ and the

goal states are $\mathcal{G} = \{s_1, \cdots, s_r\}$. The sensor can recognize goal attainment, but otherwise provides no information. Thus $\Xi(s) = \{\mathcal{S} - \mathcal{G}\}$ for all non-goal states s and $\Xi(g) = \{\mathcal{G}\}$ for all goal states g. Now, consider a modified problem $(\mathcal{S}, \mathcal{A}', \Xi', \mathcal{G})$, which has the same state space and goal set as the previous problem, but modified actions and a modified sensing function. The new sensing function Ξ' provides no information whatsoever, that is, $\Xi'(s) = \{\mathcal{S}\}$ for all states. The new actions are identical to the old, except that transitions out of goal states have been changed to self-transitions.

Consider again an execution trace in the original system from some initial knowledge state K into the goal set $\mathcal{G}$, that is $K; A_1; I_1; \cdots; A_k; I_k \subseteq \mathcal{G}$. Assuming a worst-case scenario, in which an adversary always forces non-goal transitions, the discussion from section 3.13.1 allows us to assume that the sequence of actions is a guaranteed plan for attaining the goal from K. In other words, $K; A_1; \hat{\mathcal{S}}; A_2; \hat{\mathcal{S}}; \cdots; A_k; \mathcal{S} \subseteq \mathcal{G}$, where $\hat{\mathcal{S}} = \mathcal{S} - \mathcal{G}$. Since the modified actions A_i' leave goal states invariant, we have also that $K; A_1'; \mathcal{S}; A_2'; \mathcal{S}; \cdots; A_k'; \mathcal{S} \subseteq \mathcal{G}$. In short, the modified sequence of actions is a guaranteed plan in the modified sensorless problem. Conversely, it is clear that any sequence of actions guaranteed to attain the goal in the modified sensorless problem is also a sequence of actions guaranteed to attain the goal in the original near-sensorless problem. This is because the effect of an action on a goal state is irrelevant if the goal is recognizable.

In terms of finding guaranteed strategies, we see that sensorless and near-sensorless problems are very similar. Adding a goal sensor to a sensorless problem does not change the structure of the problem much, if the applicability of the sensor depends on first executing a proper action. Conversely, for a near-sensorless problem, removing the sensor does not change the problem substantially, if the sensor can be replaced by a physical trap.

3.13.3 Probabilistic Speedup Example

Let us turn to the first example. See section 3.13.6 below, for a physical device that has important commonalities with the following example.

We will construct a non-deterministic discrete planning problem, consisting of n states and n actions. There will be one goal state, and no sensing. We will exhibit a guaranteed solution for attaining the goal from an initial knowledge state of complete uncertainty. The solution requires $2^n - n - 1$ steps, and is the shortest possible solution guaranteed to attain the goal. However, if the starting state is known exactly, there will be solutions of linear length. This suggests a guessing strategy that guesses the initial state, thus attaining the goal in quadratic expected time. Of course, one must add a goal-sensor to recognize goal attainment. However, doing so does not change the fundamental character of the problem, as one could always perform the modifications suggested in section 3.13.2. This example demonstrates that there are tasks for which randomization can speed up execution time. Furthermore, by the discussion in section 3.9, it is easy to decide whether there exists a fast randomized solution that randomizes by guessing the initial state of the system.

Let the states be $S = \{s_1, \cdots, s_n\}$, with the goal being state s_1. For convenience we will sometimes refer to states by their indices, and specify knowledge states as subsets of the integers. Thus $K = \{1, 2, 7\}$ means that the system is in one of the states s_1, s_2, or s_7, that is, $K = \{s_1, s_2, s_7\}$ in the usual notation.

The actions will have the following effect. Essentially, we want to force the system to traverse almost all knowledge states, beginning from $\{1, 2, \cdots, n\}$, before arriving at the goal $\{1\}$. Specifically, the system will be forced to first traverse all knowledge states of size $n-1$, then all knowledge states of size $n-2$, and so forth, through all knowledge states of size 2, until finally arriving at the goal $\{1\}$. Furthermore, within a collection of knowledge states of a given size, the system will be forced to traverse the knowledge states in lexicographic order. The lexicographic order of a knowledge state $K = \{s_{i_1}, s_{i_2}, \cdots, s_{i_k}\}$ (also written as $K = \{i_1, i_2, \cdots, i_k\}$) containing k elements is determined by the string $s_{i_1} s_{i_2} \cdots s_{i_k}$ of length k, where the $\{s_{i_j}\}$ are assumed to be ordered in such a way that $i_1 < i_2 < \cdots < i_k$. As an example, the knowledge state $\{2, 1, 7, 12\}$ precedes the knowledge state $\{3, 6, 1, 7\}$ since $s_1 s_2 s_7 s_{12} < s_1 s_3 s_6 s_7$ lexicographically. Observe that the first state of length k in this ordering is the knowledge state $K^k_{\min} = \{1, 2, \cdots, k\}$, whereas the last state is $K^k_{\max} = \{n-k+1, n-k+2, \cdots, n\}$. We will refer to the collection of knowledge states of size k as the *k^{th} level.*

For the sake of example, consider the case $n = 4$. The relevant knowledge states and the order in which the system will be forced to traverse them is given by the following sequence, arranged by level. Within each level the knowledge states are listed in lexicographic order from left to right

Level 4: $\{1, 2, 3, 4\}$

Level 3: $\{1, 2, 3\} \longrightarrow \{1, 2, 4\} \longrightarrow \{1, 3, 4\} \longrightarrow \{2, 3, 4\}$

Level 2: $\{1, 2\} \longrightarrow \{1, 3\} \longrightarrow \{1, 4\} \longrightarrow \{2, 3\} \longrightarrow \{2, 4\} \longrightarrow \{3, 4\}$

Level 1: $\{1\}$

The first action A_0 that we will specify is designed to permit motions between levels, specifically from the last state in each level to the first state in the next lower level, that is, from $K^k_{\max}$ to $K^{k-1}_{\min}$, for all $k = n, \ldots, 2$. In addition, A_0 should not be useful for any other motions, that is, the action should not be capable of moving the system ahead more than one knowledge state in the order that we just specified. This means that the only other motions possible should move either to a higher level or to a previous state in the same level. The action is given as:

$$
\begin{array}{rcl}
A_0: & 1 & \mapsto 1,2,\ldots,n-2,n-1 \\
& 2 & \mapsto 1,2,\ldots,n-2 \\
& \vdots & \vdots \quad \vdots \\
& k & \mapsto 1,2,\ldots,n-k \\
& \vdots & \vdots \quad \vdots \\
& n-2 & \mapsto 1,2 \\
& n-1 & \mapsto 1 \\
& n & \mapsto 1
\end{array}
$$

Since there is no sensing, we will write $A(K)$ to mean $F_A(K)$ for any action A and any knowledge state K. Observe then that indeed $A_0(K^k_{\max}) = K^{k-1}_{\min}$. Now consider an arbitrary knowledge state with k elements, say $K = \{i_1, i_2, \cdots, i_k\}$, with $i_1 < i_2 < \cdots < i_k$. Then $A_0(K) = A_0(\{i_1\}) = \{1, 2, \cdots, n - i_1\}$. If we suppose that K is not $K^k_{\max}$, then it must be the case that $i_1 < n - k + 1$. This in turn implies that $A_0(K) \supset A_0(\{n-k\}) = \{1, 2, \cdots, k\} = K^k_{\min}$. In other words, either $A_0(K)$ contains k elements and is equal to the least such set, or $A_0(K)$ contains more than k elements. In either event $A_0(K)$ appears before K in the sequence of knowledge states that we are forcing the system to traverse. Thus A_0 cannot be used to any advantage in jumping ahead in that sequence.

For the case $n = 4$, A_0 is given by:

$$
\begin{array}{rcl}
A_0: & 1 \mapsto & 1,2,3 \\
& 2 \mapsto & 1,2 \\
& 3 \mapsto & 1 \\
& 4 \mapsto & 1,
\end{array}
$$

which maps between levels as follows:

$$
\begin{array}{lll}
\text{Level 4:} & \{1,2,3,4\} & \\
& A_0 \downarrow & \\
\text{Level 3:} & \{1,2,3\} \overset{<A>}{\rightarrow} \cdots \overset{<A>}{\rightarrow} & \{2,3,4\} \\
& & A_0 \downarrow \\
\text{Level 2:} & & \{1,2\} \overset{<A>}{\rightarrow} \cdots \overset{<A>}{\rightarrow} \{3,4\} \\
& & A_0 \downarrow \\
\text{Level 1:} & & \{1\}
\end{array}
$$

Here $\overset{<A>}{\rightarrow}$ refers to any action other than A_0.

Now we must define the remaining $n - 1$ actions. The purpose of each of these

will be to permit the system to advance between consecutive knowledge states in the lexicographic ordering, while preventing the system from using the actions to advance more than one step in the ordering. In order to understand the definition of these actions, we will look at how to form the successor of a given knowledge state within a specific level, relative to the lexicographic ordering. Again, let us introduce some temporary notation. First, for the time being, whenever we write a knowledge state as a set, we will write its elements in order, so that the representation of the state corresponds to its lexicographic label. In other words a knowledge state $K = \{s_{i_1}, s_{i_2}, \cdots, s_{i_k}\}$ will be depicted in the form $K = \{i_1, i_2, \cdots, i_k\}$, with $i_1 < i_2 < \cdots < i_k$. Second, if we are only interested in the last ℓ elements of the knowledge state relative to this ordering, then we will write it as $\{\ll, i_{k-\ell+1}, i_{k-\ell+2}, \cdots, i_k\}$. In other words, the prefix "$\ll$" will mean zero or more elements whose lexicographic value is less than that of the elements that follow. If this symbol appears more than once in an equation, then it is assumed to be bound to the same value throughout the equation. And third, we will let SUCC denote the successor function relative to the lexicographic ordering and the level in which a knowledge state is located.

Now consider the successor to a knowledge state K. K is necessarily of the form $\{\ll, i_k\}$ for some i_k. If $i_k \neq n$ then $\text{SUCC}(K) = \{\ll, i_k + 1\}$. On the other hand, if $i_k = n$, then we must consider the next to last entry, that is, we must look at i_{k-1} in the representation $K = \{\ll, i_{k-1}, n\}$. Again, if $i_{k-1} \neq n - 1$ then $\text{SUCC}(K) = \{\ll, i_{k-1} + 1, i_{k-1} + 2\}$. Notice that in this case the successor function changes not only the next to last entry, but may also change the last entry. In particular, the last entry is set to be exactly one more than the next to last entry. This follows from the definition of a lexicographic order (without duplicates). Once again, if $i_{k-1} = n-1$, then we must look at the second to last entry i_{k-2}, and so forth. In general, if we are required to look at the last ℓ entries, then K must be of the form $\{\ll, i, n - \ell + 2, n - \ell + 3, \cdots, n\}$, for some i with $1 \leq i \leq n - \ell$. Thus $\text{SUCC}(K)$ is of the form $\{\ll, i + 1, i + 2, i + 3, \cdots, i + \ell\}$. The only exception to these rules is if $K = K^k_{\max}$, for some k. However, in that case, we are not interested in $\text{SUCC}(K)$ anyway, as action A_0 applies.

We will now define actions $A_1, \cdots, A_{n-1}$, where the purpose of action A_i is to change K to $\text{SUCC}(K)$ for all knowledge states of the form $K = \{\ll, i, n - \ell + 2, n - \ell + 3, \cdots, n\}$, for some ℓ. In other words, if the relevant entry in determining the successor of K has value i, then A_i will be the action that permits the system to make progress towards the goal. Furthermore, none of the other actions will permit progress at K.

From the previous discussion one sees that A_i must be of the form:

$$
\begin{array}{lrcl}
A_i: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 2 \\
 & \vdots & \vdots & \vdots \\
 & i-1 & \mapsto & i-1 \\
 & i & \mapsto & i+1 \\
 & i+1 & \mapsto & 1,2,\ldots,n \\
 & i+2 & \mapsto & i+2,i+3,\ldots,n \\
 & \vdots & \vdots & \vdots \\
 & i+j & \mapsto & i+2,i+3,\ldots,n+2-j \\
 & \vdots & \vdots & \vdots \\
 & n-1 & \mapsto & i+2,i+3 \\
 & n & \mapsto & i+2
\end{array}
$$

Notice that A_i leaves all states in the range $[1, i-1]$ unchanged. This corresponds to the "$\ll$" entries in the representation $K = \{\ll, i, n-\ell+2, n-\ell+3, \cdots, n\}$. Also, A_i advances i to $i+1$, which is the first entry changed by the successor function. State $i+1$ is non-deterministically sent to all possible states. This is done to preclude use of A_i when the relevant entry determining the successor of K actually has value $i+1$. The remaining states $i+2, \ldots, n$ are each sent non-deterministically to a subset of themselves. These sets form a tower collapsing to $i+2$, that ensures proper computation of the successor function.

We will now prove that these actions do indeed define a task for which there exists a guaranteed solution whose length necessarily is of exponential size. Then we will instantiate the actions and the strategy for the case $n = 4$.

Claim 3.17 *For the actions and task defined above, there exists a guaranteed strategy that traverses essentially all knowledge states, in the order described above. Furthermore, there is no shorter guaranteed solution.*

Proof. First, let us show that for every knowledge state containing two or more states, there is some action that makes progress towards the goal. Once we establish this, the existence of a guaranteed solution of the type described is established. Recall that progress means either moving to a successor state, or moving down to the next level, where each level consists of knowledge states of a given size.

Let $K = \{i_1, \cdots, i_k\}$, with $i_1 < \cdots < i_k$, be given. As we already indicated, if $K = K^k_{\max} = \{n-k+1, n-k+2, \cdots, n\}$, then A_0 will make progress at K. Otherwise, determine the smallest index ℓ for which K is of the form $K = \{i_1, \cdots, i_{k-\ell}, i.n-\ell+2, n-\ell+3, \cdots, n\}$, with $i_{k-\ell+1} = i$ and $1 \le i \le n-\ell$. Use $\ell = 1$ if $i_k < n$. Then action A_i will make progress at K, by construction. This follows from the following calculation (which makes use of the fact that $A_i(\{i_j\}) = i_j$

for $1 \leq i_j < i$, and the fact that $A_i(\{n-\ell+j\}) = \{i+2, i+3, \cdots, i+\ell-j+2\}$ for $2 \leq j \leq \ell$).

$$\begin{aligned} A_i(K) &= \bigcup_{j=1}^{k} A_i(\{i_j\}) \\ &= \left(\bigcup_{j=1}^{k-\ell} A_i(\{i_j\})\right) \bigcup A_i(\{i\}) \bigcup \left(\bigcup_{j=2}^{\ell} A_i(\{n-\ell+j\})\right) \\ &= \{i_1, \cdots, i_{k-\ell}\} \bigcup \{i+1\} \bigcup \{i+2, i+3, \cdots, i+\ell\} \\ &= \text{SUCC}(K). \end{aligned}$$

Now let us proceed in the other direction, and show that applying the wrong action A_i to a knowledge state cannot cause the system to advance in the ordering outlined earlier. This will establish uniqueness of the solution, in the sense that there is no shorter guaranteed strategy.

Let a knowledge state K be given, and consider applying action A_i. We have already shown that A_0 cannot make progress unless $K = K^k_{\max}$ for some k, so assume that $i > 0$. Observe that if $i+1 \in K$, then $A_i(K) = \{1, 2, \cdots, n\}$, that is, A_i maps K to complete uncertainty. This is definitely not progress, so we may as well assume that $i+1 \notin K$. Now suppose that in fact $K \subseteq \{1, 2, \cdots, i-1\}$. Then $A_i(K) = K$, which again means there is no progress. Similarly, if $K \subseteq \{1, 2, \cdots, i\}$, then $A_i(K) \subseteq \{1, 2, \cdots, i-1, i+1\}$, which is progress, but now K is of the form for which A_i was designed in the first place. So, we may assume that K intersects the set $\{i+2, \cdots, n\}$. Let ℓ be the minimal element in $K \bigcap \{i+2, \cdots, n\}$. Then $A_i(K) \supseteq A_i(\{\ell\})$. Now write K as

$$K = \left(K \bigcap \{1, \cdots, i-1\}\right) \bigcup \left(K \bigcap \{i+2, \cdots, n\}\right) \bigcup \left(K \bigcap \{i\}\right).$$

Given the minimality of ℓ, this says that $|K| = |K \bigcap \{1, \cdots, i-1\}| + |K \bigcap \{\ell, \cdots, n\}| + \chi_K(i)$, where χ_K is the characteristic function of K. Applying action A_i, we see that

$$A_i(K) = \left(K \bigcap \{1, \cdots, i-1\}\right) \bigcup \{i+2, i+3, \cdots, i+n-\ell+2\} \bigcup A_i(K \bigcap \{i\}),$$

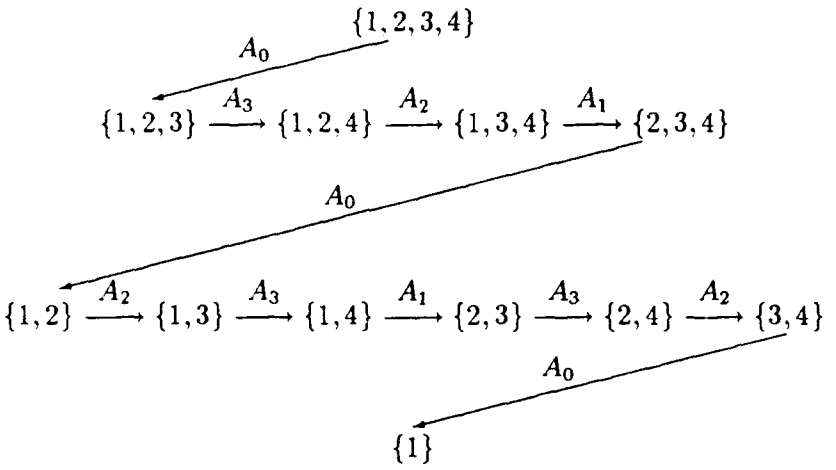
where $A_i(\{i\}) = \{i+1\}$. Thus $|A_i(K)| = |K \bigcap \{1, \cdots, i-1\}| + (n-\ell+1) + \chi_K(i)$. If $|A_i(K)| > |K|$, then A_i is moving K back up one or more levels, hence not making progress, so consider the possibility that $|A_i(K)| \leq |K|$. This is possible if and only if $n-\ell+1 \leq |K \bigcap \{\ell, \cdots, n\}|$. Clearly, this inequality can at best be an equality, in which case we must have that $K = \{\ell, \cdots, n\}$. Now there are two possibilities: either $i \in K$ or not. In the first case, we have that K is of the form $\{\ll, i, \ell, \ell+1, \cdot\ \cdot, n\}$, in which case A_i is designed to make progress at K. Thus, finally, assume that $i \notin K$. So, $K = \{\ll, \ell, \ell+1, \cdots, n\}$ and $A_i(K) = \{\ll, i+2, \cdots, i+n-\ell+2\}$, with $i+2 \leq \ell$.

But this says that either $A_i(K)$ is equal to K or $A_i(K)$ precedes K lexicographically. In short, A_i does not make progress at K. ∎

Let us instantiate these actions for the case $n = 4$. We have

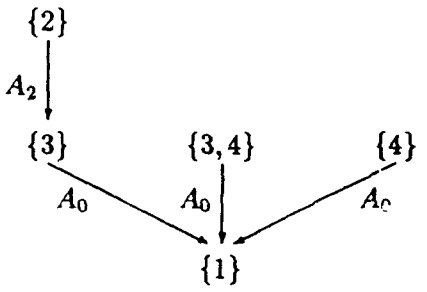
$$
\begin{array}{lll}
A_1: & 1 \mapsto 2 & \\
& 2 \mapsto 1,2,3,4 & \\
& 3 \mapsto 3,4 & \\
& 4 \mapsto 3, &
\end{array}
\qquad
\begin{array}{lll}
A_2: & 1 \mapsto 1 & \\
& 2 \mapsto 3 & \\
& 3 \mapsto 1,2,3,4 & \\
& 4 \mapsto 4, &
\end{array}
\qquad
\begin{array}{lll}
A_3: & 1 \mapsto 1 & \\
& 2 \mapsto 2 & \\
& 3 \mapsto 4 & \\
& 4 \mapsto 1,2,3,4. &
\end{array}
$$

The guaranteed solution is given by:



We see then that there are tasks for which the planning and execution times are exponential in the size of the input. Observe, however, for this particular example, that if the initial state of the system were known precisely, then there would be a fast solution for attaining the goal. In particular, if the initial state is s_1 then the system is already in the goal. If the initial state is either s_3 or s_4, then action A_0 will attain the goal in a single motion. Finally, if the initial state is s_2, then action A_2 will cause a transition to state s_3, from which A_0 will attain the goal. In short, if one writes out the dynamic programming table to two columns for this task, then one has a collection $\{K_i\}$ of knowledge states that cover the entire state space. Thus one can employ a randomized strategy that guesses the initial state of the system, then executes a short sequence of actions designed to attain the goal. One must, of course, add a goal sensor, in order to ensure reliable goal recognition.

For the sake of completeness, note that the relevant portion of the backchaining diagram corresponding to the dynamic programming table out to two columns is given by the following diagram (depicting vertical levels rather than horizontal columns):



In the general case, one must backchain out to the $n - 2^{nd}$ column of the dynamic programming table. A guessing strategy consists of guessing between the $n-1$ non-goal states, then executing a strategy of no more than $n-2$ steps, that is guaranteed to attain the goal if the guess is correct. Thus the expected number of actions executed until the goal is attained is on the order of n^2.

Notice that adding a goal sensor does not fundamentally change the exponential character of the guaranteed strategy, by the partial equivalence of sensorless and near-sensorless tasks, as established in section 3.13.2. It is important to keep this partial equivalence in mind, since a goal sensor clearly permits a speedup of the guaranteed solution if one does not make the modifications suggested by the partial equivalence. We thus have the following claim.

Claim 3.18 *There exists a near-sensorless discrete planning problem in which the shortest guaranteed strategy has exponential length, but for which there exists a randomized strategy that only requires quadratic expected time.*

Proof. Most of this claim has been proved. We only need to verify that there does indeed exist a linear time strategy for attaining the goal if the initial state of the system is known. We return to the construction above.

First notice that action A_0 is guaranteed to move state s_n and s_{n-1} into the goal in a single motion. Observe also that action A_i is guaranteed to move state s_i to state s_{i+1} for all i. This establishes the claim. ∎

In retrospect, the randomizing part of the claim is not very surprising. The actions A_i are actually fairly deterministic. However, the solutions are not at all commensurate. Said differently, the solution for a given initial state is not guaranteed tc serendipitously make progress at other states. This is quite unlike the fortunate situation that we encountered with one-dimensional random walks, where the same solution pretty much applied to all possible states. Thus the surprising aspect of the claim is the exponential character of the guaranteed solution for what may seem to be fairly deterministic actions.

3.13.4 An Exponential-Time Randomizing Example

The following example exhibits a (near-)sensorless task for which the shortest guaranteed solution requires an exponential number of steps and for which a

randomized solution that guesses the starting state also requires exponential time. The basic idea is to generate a problem in which the knowledge states play the role of bit vectors, that may be modified only by counting.

The example will consist of n states, and $2n-3$ actions. We will present the example as if there is no sensing, bearing in mind the partial equivalence between sensorless and near-sensorless problems of section 3.13.2. We retain some of the notation from the previous example (section 3.13.3). In particular, we will interchangeably refer to a state either as s_i or as i, for $i = 1, \ldots, n$.

The state space will be of the form $\mathcal{S} = \{s_1, \cdots, s_n\} = \{1, \cdots, n\}$, with the goal being state s_1. We will denote the actions by the symbols $A_1, \cdots, A_{n-1}$ and $B_1, \cdots, B_{n-2}$. We will write knowledge states as ordered tuples, as we did in the previous section. In other words, a knowledge state K of size k will be written in the form $K = \{s_{i_1}, \cdots, s_{i_k}\} = \{i_1, \cdots, i_k\}$, with $i_1 < \cdots < i_k$. Thinking of a knowledge state as a bit vector, K will correspond to the number $x(K)$, with

$$x(K) = \sum_{i \in K} 2^{n-i}.$$

Conversely, given an integer x in the range $[0, 2^n - 1]$, there is a unique knowledge state K for which $x(K) = x$. We will denote this knowledge state by $K(x)$, with

$$K(x) = \{i \mid \text{bit \# } (n-i) \text{ is a 1 in the binary representation of } x\}.$$

As an example, if $n = 10$ and $K = \{1, 3, 7\}$, then $x(K) = 648$. Similarly, if $n = 4$ and $x = 9$, then $K(x) = \{1, 4\}$.

As before, we will let the prefix symbol "$\ll$" in the representation $K = \{\ll, i_1, \cdots, i_\ell\}$ denote zero or more elements whose lexicographic order precedes that of i_1. This notation carries over to the binary representation of the number $x(K)$. Comparing the binary representation of $x(K)$ with K, we have the following schematic:

Bit #:	$\cdots$		$\cdots$	$\underbrace{n-i_1}$	$\cdots$	$\underbrace{n-i_2}$	$\cdots$	$\underbrace{n-i_\ell}$	$\cdots$	0
$x(K)$:	$\ll$	0 $\cdots$ 0		1	0 $\cdots$ 0	1	0 $\cdots$ 0	1	0 $\cdots$	0
				$\Updownarrow$		$\Updownarrow$		$\Updownarrow$		
K:	$\{\ll,$			$s_{i_1},$		$s_{i_2},$	$\cdots$	$s_{i_\ell}\}$		

The actions that we will construct will force the system to traverse an exponential number of knowledge states, beginning with the state of complete uncertainty $\{1, \cdots, n\}$, and ending with the goal state $\{1\}$, in an order that corresponds to counting downwards from $2^n - 1$ to 2^{n-1}. For the special case $n = 4$, this corresponds to the following transitions (for later reference the transitions are also labelled with the associated actions):

K	$x(K)$	(actions)
$\{1,2,3,4\}$	15	
$\downarrow$	$\downarrow$	A_3
$\{1.2,3\}$	14	
$\downarrow$	$\downarrow$	B_2
$\{1,2,4\}$	13	
$\downarrow$	$\downarrow$	A_2
$\{1,2\}$	12	
$\downarrow$	$\downarrow$	B_1
$\{1,3,4\}$	11	
$\downarrow$	$\downarrow$	A_3
$\{1,3\}$	10	
$\downarrow$	$\downarrow$	B_2
$\{1,4\}$	9	
$\downarrow$	$\downarrow$	A_1
$\{1\}$	8	

Let us first define the actions $\{A_k\}$. These are designed to count down from knowledge states K whose associated numbers $x(K)$ are odd. Since an odd number contains a one in the least significant bit, the knowledge state must contain the state s_n. The actions $\{A_k\}$ are designed to remove this state. We have, for $k = 1, \ldots, n-1$,

$$
\begin{array}{rrcl}
A_k: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 2 \\
 & \vdots & \vdots & \vdots \\
 & k & \mapsto & k \\
 & k+1 & \mapsto & 1,2,\ldots,n \\
 & \vdots & \vdots & \vdots \\
 & n-1 & \mapsto & 1,2,\ldots,n \\
 & n & \mapsto & k.
\end{array}
$$

[Note, of course, that if $k = n-1$, then $A_k: n-1 \mapsto n-1$.]

Similarly, the actions $\{B_k\}$ are designed to count down by one from knowledge states whose associated numbers are even. Thus these actions must worry about borrowing properly from higher order bits. We have, for $k = 1, \ldots, n-2$,

$$
\begin{array}{rrcl}
B_k: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 2 \\
 & \vdots & \vdots & \vdots \\
 & k & \mapsto & k \\
 & k+1 & \mapsto & k+2, k+3, \ldots, n \\
 & k+2 & \mapsto & 1, 2, \ldots, n \\
 & \vdots & \vdots & \vdots \\
 & n & \mapsto & 1, 2, \ldots, n.
\end{array}
$$

For the special case $n = 4$, we have the following five actions:

$$
\begin{array}{rrcl}
A_1: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 1,2,3,4 \\
 & 3 & \mapsto & 1,2,3,4 \\
 & 4 & \mapsto & 1,
\end{array}
\qquad
\begin{array}{rrcl}
A_2: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 2 \\
 & 3 & \mapsto & 1,2,3,4 \\
 & 4 & \mapsto & 2,
\end{array}
\qquad
\begin{array}{rrcl}
A_3: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 2 \\
 & 3 & \mapsto & 3 \\
 & 4 & \mapsto & 3,
\end{array}
$$

$$
\begin{array}{rrcl}
B_1: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 3,4 \\
 & 3 & \mapsto & 1,2,3,4 \\
 & 4 & \mapsto & 1,2,3,4,
\end{array}
\qquad
\begin{array}{rrcl}
B_2: & 1 & \mapsto & 1 \\
 & 2 & \mapsto & 2 \\
 & 3 & \mapsto & 4 \\
 & 4 & \mapsto & 1,2,3,4
\end{array}
$$

Claim 3.19 *For the actions and task defined above, there exists a guaranteed strategy that traverses essentially all knowledge states, in the order described above. Specifically, the strategy traverses all knowledge states that contain state s_1. There are 2^{n-1} such knowledge states. Furthermore, there is no shorter guaranteed solution.*

Proof. First, let us show that for every knowledge state K there is some action that makes progress. In this case progress means that the number determined by the bit-vector representation of K is decreased. In fact, we will exhibit an action that decreases $x(K)$ by exactly one.

Suppose that $x(K)$ is odd. Let k be the order of the least significant bit other than bit #0 which is set to 1. Then

$$x(K) = \text{«} 1 \overbrace{0 \cdots 0}^{k-1} 1,$$

meaning that $K = \{\text{«}, s_{n-k}, s_n\}$. Now note that $A_{n-k}(K) = \{\text{«}, s_{n-k}\}$, so $x(A_{n-k}(K)) = x(K) - 1$, as desired.

On the other hand, suppose that $x(K)$ is even. Again, let k be the order of the least significant bit that is set to 1. Then $k \geq 1$, and

$$x(K) = \ll 1\overbrace{0\cdots 0}^{k}.$$

If $y = x(K) - 1$, then

$$y = \ll 0\overbrace{1\cdots 1}^{k}.$$

This says that $K = \{\ll, s_{n-k}\}$ and that $K(y) = \{\ll, s_{n-k+1}, s_{n-k+2}, \cdots, s_n\}$. Now note that $B_{n-k-1}(K) = K(y)$, as desired.

We have shown that, for any knowledge state K, there is a strategy for counting down from $x(K)$. In particular, suppose $K = \{i_1, \cdots, i_\ell\}$, with $i_1 < \cdots < i_\ell$. If $i_1 = 1$, then one can count from $x(K)$ down to 2^{n-1}, at which point the goal is attained. On the other hand, if $i_1 > 1$, then one can count down from $2^{n-1} + x(K)$ to 2^{n-1}, at which point the goal is attained. This amounts to pretending that $s_1 \in K$. Alternatively, one could just count down from $x(K)$ to 1, which places the system in state s_n. Applying action A_1 then attains the goal. If one looks at the details, these two strategies are really the same. After all, the counting never involves changing the bit corresponding to s_1.

Second, we must show that applying the wrong action at a knowledge state cannot make further progress. This will establish that the strategy just outlined is the shortest strategy guaranteed to attain the goal.

So, suppose that knowledge state K is given, and let $x = x(K)$.

Consider applying action A_k, for some k. If $K \cap \{s_{k+1}, \cdots, s_{n-1}\} \neq \emptyset$ then $A_k(K) = \{s_1, \cdots, s_n\}$, which is certainly not progress. On the other hand, if $K \subseteq \{s_1, \cdots, s_k\}$, then $A_k(K) = K$, which again is not progress. That leaves the possibility that $K \subseteq \{s_1, \cdots, s_k\} \cup \{s_r\}$. Suppose that both $s_k \in K$ and $s_n \in K$. Then A_k is designed to make progress at K, so that's fine. On the other hand, suppose that $s_k \notin K$ and $s_n \in K$. Then $K = \{\ll, s_n\}$, while $A_k(K) = \{\ll, s_k\}$. Note that $x(A_k(K)) > x(K)$, so this motion also does not make progress.

Consider applying action B_k, for some k. If $K \subseteq \{s_1, \cdots, s_k\}$, then $B_k(K) = K$, which means no progress. If K contains any elements from the set $\{s_{k+2}, \cdots, s_n\}$, then $B_k(K)$ is the entire state space, that is, complete uncertainty. The remaining case says that $K = \{\ll, s_{k+1}\}$, but then K is of the form for which B_k was designed to make progress. ∎

Observe that the previous proof also shows that if the state of the system is known exactly, say $K = \{s_i\}$, then the only reasonable strategy for attaining the goal is to count down to 1 from 2^{n-1}, followed by an application of action A_1. This is because applying the wrong action at a knowledge state essentially has one of two effects: Either (1) the action does not change the knowledge state, or (2) the action yields complete uncertainty. The exception to this rule is given by the effect on state s_n, but this state lies one action away from the goal, and misapplying an action when the system is in state s_n only moves it further away.

Thus we have

Claim 3.20 *There exists a near-sensorless discrete planning problem in which the shortest guaranteed strategy has exponential length. Furthermore, the expected running time of any randomized strategy is also exponential in the number of states and actions.*

3.13.5 Exponential-Sized Backchaining

The following example demonstrates that there are sensorless tasks for which the dynamic programming approach of backchaining can generate a table of exponential size even if one only backchains a linear number of steps. In fact we will exhibit an example with n states and $n^2 - n$ actions in which the knowledge state $\mathcal{S}$ is obtained in the $n-1^{st}$ column of the dynamic programming table, and in which an exponential number of knowledge states are generated in between. Of course, this implies that there exists a fast strategy for attaining the goal. Indeed, there is a linear-time strategy. Furthermore, it may be possible to arrive at that strategy quickly, by using an approach other than the dynamic programming approach. For our particular example all actions will be deterministic. This immediately says that there is a fast planning algorithm, using Natarajan's graph-searching techniques (see [Nat86]). However, one can easily modify the actions so that they are non-deterministic. In short, this example says nothing about the fundamental complexity of planning under uncertainty, merely something about planning using backchaining. More fundamental results are contained in [Pap] and [PT], as we have already mentioned.

The state space is $\mathcal{S} = \{s_1, \cdots, s_n\} = \{1, \cdots, n\}$. The $n^2 - n$ actions are given by:

$$A_{ij}(s_k) = \begin{cases} s_i, & \text{if } k = j \\ s_k, & \text{otherwise.} \end{cases} \qquad 1 \leq i, j \leq n,\ i \neq j.$$

In other words, A_{ij} collapses the two states s_i and s_j to the state s_i, while leaving all other states invariant. There is no sensing.

We will start the backchaining process off by assuming that any singleton state is a goal. In other words, if the system can unambiguously move into some single state, then it has achieved its goal. It is easy to change this problem into one in which the system must attain a particular goal state, by adding an action and a state to the construction. In any event, we may assume that column number zero of the dynamic programming table contains entries for all knowledge states of the form $\{k\}$, for $k = 1, \cdots, n$.

Now suppose that the planner is backchaining from the ℓ^{th} column of the dynamic programming table, and that all the non-blank entries in this column are of size at most $\ell + 1$. Suppose further that the collection of non-blank entries includes all knowledge states of size $\ell + 1$. Since no action collapses more than two states, it is impossible to obtain knowledge states of size greater than $\ell + 2$ in the $\ell + 1^{st}$ column. However, it is possible to obtain all knowledge states of size $\ell + 2$. This says that precisely in column number $(n - 1)$ the knowledge state $\mathcal{S}$ will have its entry filled

in for the first time. Furthermore, all other knowledge states will also have had their entries filled in.

3.13.6 The Odometer

The following physical device has important commonalities with the graph example presented in section 3.13.3. In particular, the task described by this device has a guaranteed solution that requires an exponential number of steps, and a randomized solution that only requires an expected linear number of steps.

Imagine a series of n horizontal plates or wheels arranged vertically above each other. The plates are connected by a gearing mechanism that acts much like an odometer. Specifically, a primitive action consists of turning a plate one-tenth of a revolution. Call this a *partial turn*. Whenever a given plate turns, it also turns the plate above it, but at one tenth the speed, so each time a plate makes one full revolution, the plate immediately above makes a partial turn. Similarly, turning a plate turns the plate directly below it at ten times the speed. There is a crank below the bottom plate which turns that plate, and consequently all other plates at reduced speeds. Under certain circumstances mentioned later, individual plates may also be turned directly. The crank and any individual plate can only be turned at a specific fixed speed, say, one partial turn per unit time. (Turning an individual plate directly also turns the other plates via the gearing mechanism, as described earlier.)

On one of the plates is a ball. The ball arrives from a distribution bin which non-deterministically places the ball on a non-deterministically chosen plate. There is a chute next to each plate. Turning the plate so that the ball passes by this chute causes the ball to roll off the plate, down the chute, and onto the plate below. The chutes are themselves arranged in unison above each other. They are hinged to a vertical pole, and may be swung away from the plates. In this case, if a plate is turned so that the ball passes by the location at which the chute would normally be, the ball simply drops vertically. If the ball is not caught by someone, it reenters the distribution bin and is once again non-deterministically placed on a plate. The plates cannot be turned individually when the chutes are in place; only the crank may be used. However, the plates may be turned individually when the chutes have been swung away from the plates.

There are thus two ways to remove a ball from a plate. The first is to swing the chutes away from the plates, move one's hand up to the plate containing the ball, then turn the plate until the ball falls out and onto one's hand. The second way is to swing the chutes into place, then turn the crank until the ball emerges from the bottom plate.

The first approach requires turning the given plate at most 10 partial turns before the ball falls out. The second approach may require turning the crank as many as $\frac{10}{9}(10^n - 1)$ partial turns, should the ball happen to be on the top plate at the start. Clearly, assuming that one can determine on which plate the ball is resting, the first approach is preferable.

Now, suppose, however, that one cannot determine on which plate the ball is

resting. For instance, the plates might be covered. Then the only guaranteed strategy for removing the ball is to turn the crank with the chutes in place, until the ball emerges. Turning any individual plate, with the chutes swung away, runs the risk of causing the ball to drop from a plate, forcing it back into the distribution bin. From a worst-case point of view, that strategy might never terminate. Consequently, the only guaranteed strategy may require a long time to execute.

Fortunately, a randomized solution consists of guessing the plate on which the ball is resting, then acting as if that plate did indeed hold the ball. In other words, in the absence of a sensor, the randomized strategy simulates one. With probability $1/n$, the strategy will pick the correct plate. If it picks the wrong plate, then the ball is repositioned, and the strategy can try again. The expected number of partial turns until the ball emerges is thus bounded by $10n$. This is only a linear factor more than in the case in which a sensor is available, well below the exponential guaranteed strategy.[5]

3.14 Summary

This chapter considered the problem of planning in the presence of uncertainty in discrete spaces. The standard dynamic programming approach was extended to include an operator that would purposefully make randomizing choices. The motivation for including this operator was to extend the class of solvable tasks beyond those solvable by guaranteed strategies. Not all tasks admit to what traditionally are considered guaranteed solutions. These are solutions that are certain to accomplish their tasks in a fixed and bounded number of run-time operations that may be ascertained at planning time. There are many tasks that one would consider solvable simply because they may be accomplished frequently even if not always. By placing a loop around a strategy that tries to solve such a task, one can often be certain of a solution eventually. Although in principle the solution could require an unbounded amount of time, often one may be able to compute the expected time until the task is solved. In particular, by purposefully randomizing its decisions a strategy can sometimes enforce a minimum probability of success on any particular attempt, thereby placing an upper bound on the expected time until task completion.

The basic scheme is to compute partial plans that are guaranteed to accomplish portions of the task. Generally these partial plans will only succeed if fairly stringent initial conditions are satisfied. While any particular plan's preconditions may not be satisfiable, the union of all the preconditions may be satisfiable. This means that in fact some partial plan's preconditions are satisfied, but due to uncertainty the system cannot ascertain which plan's preconditions. In that case it makes sense to guess the appropriate partial plan. Effectively the strategy is executing a randomizing action by guessing which partial plan is applicable. If the guess is correct, then the task will

[5]Of course, the randomized strategy may require more than the expected number of trials to succeed on any particular execution. However, the probability of requiring several factors of this expectation decreases exponentially quickly in the number of factors.

be accomplished. Otherwise, the system will need to guess again, until it eventually accomplishes the task.

Of particular interest were simple feedback loops. Theses are strategies that only consider current sensed values in deciding on motions to execute. Such strategies are often useful when there is some progress measure on the state space that measures the system's distance from task completion. Whenever possible, the system will execute an action that makes progress relative to the progress measure. Otherwise, the system will execute a randomizing motion. The purpose of the randomizing motion is to either accomplish the task or move to some location from which the available sensory information again permits progress. In this context the chapter explored various types of random walks. It was shown that if the expected speed of progress is uniformly bounded away from zero, then it is possible to bound the expected time until task completion. The bound is the intuitively desirable bound of distance divided by expected velocity, where distance is defined by the progress measure.

Chapter 4

Preimages

In this and the next chapter we will turn our attention to continuum spaces, primarily spaces such as $\Re^n$. The same ideas that appeared in the chapter on discrete planning problems will appear in the context of continuous planning problems. In particular, the notions of expected progress and randomization by guessing starting states will carry over naturally and prove useful. Rather than develop the whole framework afresh, we will focus on particular examples and results that should make the connection between the continuous and discrete cases clear.

4.1 Preimage Planning

In the chapter on discrete planning problems, planning with uncertainty was viewed as planning in the space of knowledge states. This view effectively reduced the problem of finding a guaranteed strategy in a space with both imperfect control and imperfect sensing to a backchaining problem in a space with imperfect control and perfect sensing. Backchaining was implemented by dynamic programming, using a boolean cost function. A similar approach applies in continuous spaces. The preimage planning approach developed by [LMT] formally introduced this notion into robotics. We will briefly review this approach in this section. The domain will be taken to be the configuration space of the robot or part being moved relative to whatever obstacles there may be in the environment (see [Loz83]). We will, however, often restrict ourselves to $\Re^n$, for some n, with polyhedral obstacles. This might correspond to the configuration space of either a cartesian robot or a polyhedral part which is only permitted to translate but not to rotate.

Uncertainty

First let us define uncertainty. We have already indicated that sensing errors are modelled as bounded error balls. Thus, if the system is in state $\mathbf{x}$ at execution time, then the position sensor may return a value $\mathbf{x}^*$ that lies within some distance ϵ, of $\mathbf{x}$. In the language of chapter 3, once we postulate full sensing consistency, then the collection of possible sensory interpretation sets is given by the collection of balls

$\{B_{\epsilon_s}(\mathbf{x}^*)\}$, as $\mathbf{x}^*$ varies over $B_{\epsilon_s}(\mathbf{x})$. If the sensors are more complicated than this, in particular if there are sensors that measure other attributes of the system, such as velocity or force, then one can model this by increasing the dimensionality of the state space of the system to include these other attributes. Alternatively, if the future state of the system does not depend on these attributes, then one need not raise the dimensionality of the planning space. Instead, one can model the additional sensing information by projecting it into the original state space. For example, the measured force may indicate that the object is in contact with some surface S. This generally reduces the position sensing uncertainty, by selecting a lower dimensional slice of the sensing error ball, corresponding to the intersection of the surface with the position interpretation, that is, $S \cap B_{\epsilon_s}(\mathbf{x}^*)$. There are some subtleties here. For instance, the interpretation of a force or a velocity may depend on the action executed. This means that possible sensory interpretation sets must now be modelled not only as functions of the state of the system, but also as functions of the commanded action. While we did not model this dependence in the discrete setting, doing so does not pose any fundamental difficulties. Having said all this, we will basically ignore sensing of attributes other than position in our examples. For more detailed investigations of sensing in the context of preimage planning see [LMT], [Mas84], [Erd84], [Buc], [Don89], [Can88], [Lat], among others.

Control uncertainty is defined similarly. At execution time, whenever a nominal control command is issued, the actual effect on the system is given by a range of effective commands that lie in some error ball about the nominal command. More general models of control uncertainty are of course possible. Within the **LMT** preimage methodology, the envisioned commands are either applied forces or applied velocities. In fact, **LMT** focuses on an equivalence between forces and velocities given by modelling dynamics as generalized damper dynamics, an assumption that produces a first-order system. Specifically, control commands are nominal velocities $\mathbf{v}_0$; the evolution of the system is governed by the first-order equation

$$\mathbf{F} = \mathbf{B}(\mathbf{v} - \mathbf{v}_0^*), \tag{4.1}$$

where $\mathbf{v}$ is the actual velocity of the system, $\mathbf{F}$ is the force exerted by the environment on the system, $\mathbf{B}$ is a *damping matrix*, and $\mathbf{v}_0^*$ is the effective commanded velocity. The damping matrix is often simply taken to be the identity matrix, perhaps multiplied by some gain factor. Control uncertainty is represented by the term $\mathbf{v}_0^*$. This is assumed to lie in some error ball $B_{\epsilon_v}(\mathbf{v}_0)$ about the nominal commanded velocity. See figure 4.1. It is sometimes convenient to think of the velocity error as defining an error cone. This cone represents the trajectories that can locally emanate from a given point.

Generalized damper dynamics are convenient, since they model the (error-free) trajectories of the system as piecewise linear motions. Similarly, in the presence of uncertainty, the possible trajectories may be modelled as cones. For further discussion on generalized damper dynamics see [Whit77] and [LMT]. We will henceforth assume that the dynamics are generalized damper dynamics in $\Re^n$, with polyhedral obstacles.

Observe that these models of uncertainty are bounded worst-case models. In

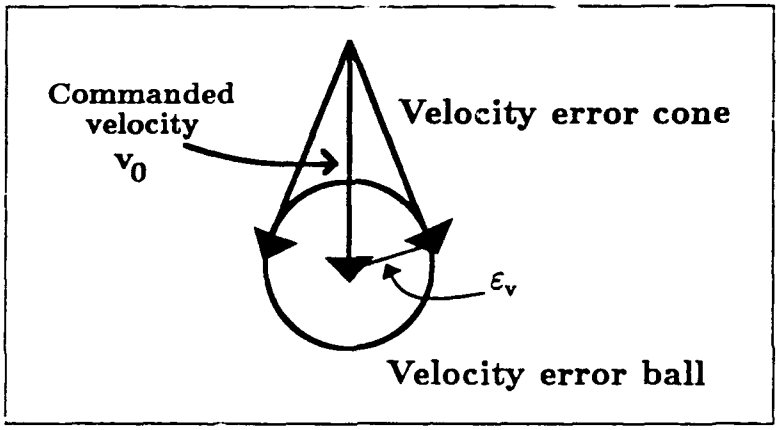


Figure 4.1: Velocity error ball about a nominal velocity command. If one is only interested in directions, sometimes it is useful to think of the error as an error-cone.

other words, nothing is said about the actual distribution of sensor values or control commands within the uncertainty balls. The distributions may be probabilistic, they may be fixed biases, or they may even be chosen in a worst-case manner by an adversary.

For future convenience we will also assume that the sensing and control error balls are all open balls.

Preimages and Termination Predicates

Integral to the planning of guaranteed strategies is the notion of a *preimage*. Intuitively, a preimage of a collection of goals is a region in state space from which a certain action is guaranteed to attain one of the goals, and do so in a recognizable manner. Goals are themselves modelled geometrically as regions in state space. Forming the preimages of a goal is analogous to backchaining one column in the dynamic programming table. However, it is not exactly the same thing. Into the definition of a preimage enters the notion of a *termination predicate*. The termination predicate is the decision process that terminates a motion at run-time, signalling goal attainment. The amount of information that a termination predicate considers determines the power of the planning system to solve certain tasks. Essentially, the termination predicate performs the forward projection of states and the intersection with sensory interpretation sets discussed in the discrete setting. If a termination predicate considers only current sensed values in deciding goal attainment, then, in the terminology of chapter 3, one has a planning problem involving strategies that are simple feedback loops. If the termination predicate considers all possible past sensed values as well as time-indexed forward projections then one has a planning problem analogous to the full dynamic programming approach discussed in the discrete setting. There are numerous intermediate possibilities, some of which did not

seem as evident in the discrete case. One important variation is to forward project the start region under a given commanded velocity, but then to use only current sensor values intersected with this forward projection in determining goal attainment. See [Erd86]. See also page 209 for further discussion of termination predicates.

Knowledge States

One important characteristic of the termination predicates employed in the LMT framework is their Markovian nature. This means that the entire information available to a termination predicate at any given time may be summed up in a single set describing the possible configurations of the system. In the discrete setting this set was referred to as a knowledge state. The existence of such a knowledge state assumes that the state space of the system is Markovian as well, that is, that the future behavior of the system depends only on the current state of the system and the action being executed. It also assumes, as we have been throughout the thesis, that the sensor values obtained at execution time depend only on the current state of the system. An implication of this observation is that a termination predicate can forget the exact sensor values and forward projections that gave rise to the current knowledge state. Equivalently, supplying a termination predicate with a given knowledge state and starting a motion from anywhere inside the set of configurations described by that knowledge state permits the termination predicate to make precisely the same decisions that it would have made if it had encountered the same knowledge state during a motion that had originated from some other region at some prior time. See [Mas84] for a description of how a termination predicate functions.

Actions and Time-Steps

One aspect may be troubling in comparing the discrete and continuous settings. In the continuous setting it seems that one always needs a termination predicate to stop a motion. After all, the basic commands are velocities, so one needs some form of termination to switch between different velocities. In contrast, in the discrete setting, termination predicates were never explicitly required. Instead, each step involved some action, which terminated by definition, whereupon the available sensory information was used to select a new action. In fact, the analogy between the discrete and continuous settings becomes apparent if one considers actions to be velocities executed over some duration of time. In particular, velocities executed over infinitesimal time, or over the cycle time of the control loop, form the natural analogue in the continuous case of the single-step actions in the discrete case. Conversely, a velocity executed until some termination predicate signals goal attainment has as counterpart in the discrete setting a repeated application of the same action until some condition that is a combination of sensory information and history is satisfied.

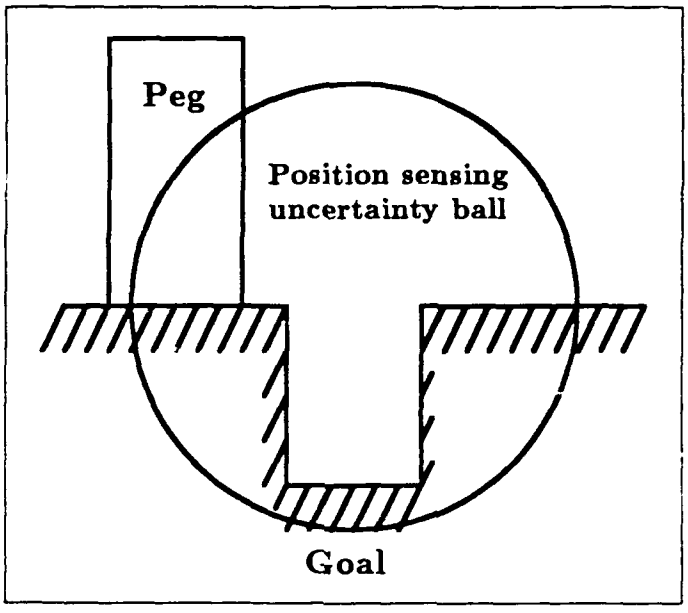


Figure 4.2: The task is to slide the peg into the hole. Given large position sensing uncertainty, a simple feedback loop that does not remember its past state will become confused near the hole.

History

Notice that once one establishes that primitive actions are really velocities executed over a small duration of time, then the notion of a simple feedback loop makes sense both in the discrete and continuous cases. It is simply a control loop in which at each instant in time the command issued depends only on the current sensed values. In contrast, the notion of a preimage which employs an action over an extended period of time tacitly includes some history. This history may simply be the information implicit in knowing that a termination predicate will eventually signal success. As an example, consider the task of sliding a peg into a hole, as in figure 4.2. If position sensing uncertainty is large, then the system cannot know which side of the hole the peg is on once it is near the hole. Thus a simple feedback loop would have to resort to randomization as used in the example of section 2.4. On the other hand, if the system is far enough away from the hole, then it can decide which way to move. Having chosen a motion direction, and a termination predicate that recognizes goal attainment by noting that the peg is falling into the hole, the system can proceed to move in the correct direction, ignoring all sensor values except the final one that signals goal attainment. In short, there are two preimages, corresponding to being far enough to the left or right of the hole. And although it is true that the termination predicate does not need history to recognize goal attainment, the strategy employs history in knowing that certain sensor values are irrelevant. The history is implicitly

used to rule out the confusion that a simple feedback loop would encounter. This is an important distinction, which makes clear that a preimage in the continuous setting corresponds to a special type of strategy with history in the discrete setting. In particular, a preimage is a strategy that locally is guaranteed to make progress towards the goal.

Preimages: Definition

The preimage R relative to a commanded velocity $\mathbf{v}_0$ of a collection of goals $\{G_\alpha\}$ is specified implicitly as the solution to an equation of the form

$$P_{\mathbf{v}_0,R}(\{G_\alpha\}) = R.$$

Here the operator $P_{\mathbf{v}_0,R}$ defines a subset of the region R from which recognizable goal attainment is guaranteed. Recognizable attainment means that the termination predicate will successfully halt the motion, specifying which goal G_α has been attained. The termination predicate is given the start region R as data, and may use this data in deciding whether the goal has been attained. Of course, the termination predicate need not use R. For instance, if the termination predicate being employed only considers current sensed values, then it would ignore the start region R. See [LMT], [Mas84], [Erd86], and [Don89] for further details on the specification of the preimage equation. We will content ourselves here with this brief explanation, bearing in mind the planning approach discussed in the chapter on discrete planning problems.

Planning by Backchaining

Planning a guaranteed strategy consists of backchaining preimages, much like in the dynamic programming approach. The analogy in the discrete setting would be to backchain several substrategies each of which makes progress locally until some subgoal is attained. In the continuous case the formal definition proceeds as follows (see [LMT] and [Mas84] for further details). Let $\mathcal{G}_0 = \{G_\alpha\}$ be the collection of task-level goals. Now, suppose that $\mathcal{G}_k$ is defined as some collection of subgoals to be attained. One backchains by forming all preimages $R_{\beta,k+1}$, which satisfy the preimage equation $P_{\mathbf{v}_0,R_{\beta,k+1}}(\mathcal{G}_k) = R_{\beta,k+1}$, for some commanded velocity $\mathbf{v}_0 = \mathbf{v}_0(R_{\beta,k+1})$ that depends on the actual preimage. This collection of preimages forms the collection of subgoals for the next level of backchaining, that is, $\mathcal{G}_{k+1} = \{R_{\beta,k+1}\}_{\beta\in B}$, where B is some appropriate index set. Planning either stops when some preset limit on k has been reached, or when no further preimages can be computed. The task is said to be solvable if the initial knowledge state of the system $\mathcal{I}$ is contained in some preimage generated during this backchaining process. $\mathcal{I}$ is a subset of the state space that is known to contain the actual initial state of the system. Executing a strategy entails collapsing this recursion, just as in the discrete case. In other words, given that the system is in a preimage $R_{\beta,k} \in \mathcal{G}_k$ at the k^{th} level, the system executes action $\mathbf{v}_0(R_{\beta,k})$ until some subgoal $R_{\beta',k-1}$ in the $k-1^{st}$ level $\mathcal{G}_{k-1}$ is attained. This process is repeated until a task goal G_α is attained. We refer to such a strategy as a *guaranteed strategy*

since it is certain to attain a task goal in a specific number of steps. This stands in contrast to a randomized strategy, which only has some probability of attaining a task goal and thus may fail to solve a task in any fixed number of steps.

4.2 Guessing Strategies

With these preimage definitions in hand, one can now define the guessing operator SELECT for the continuous case. For the case of initial-state-guessing this amounts to backchaining preimages until one has a collection $\{R_{\beta,k}\}_{\beta \in B}$ at the k^{th} level that covers the initial state of the system $\mathcal{I}$. A randomized strategy consists of randomly selecting one of these $R_{\_,k}$ as the guessed starting region, then executing the guaranteed strategy for $R_{\beta,k}$. The strategy is a guaranteed strategy for attaining a task level goal, in the sense that the strategy would reliably and recognizably attain one of the G_α if the system knew for certain that its starting state was in the preimage $R_{\beta,k}$. However, the starting state is merely guessed, and thus the usual admonishments regarding reliable goal recognition and reliable restart of the strategy apply [see section 3.9 for the discrete case]. For this reason we will assume, as we did in the discrete case, that the task-level goals $\{G_\alpha\}$ are recognizable. This means that if the system is ever in one of the sets G_α it will know so based purely on current sensing and not on the history of the motion. Similarly, we will assume that the system never strays out of some region $\mathcal{X}$, where $\mathcal{I} \subseteq \mathcal{X} \subseteq \bigcup_{\beta \in B} R_{\beta,k}$. In other words, the sets $\{R_{\beta,k}\}_{\beta \in B}$ may used repeatedly for restarting the guessing loop.

The discussion of randomized strategies that guess the initial state of the system generalizes to the more general case of randomized strategies that make multiple guesses, much as discussed in section 3.11 for the discrete case.

4.2.1 Ensuring Convergence of SELECT

A more serious issue is whether the operator SELECT is meaningful in the continuous setting. Cause for concern stems from the possible infinite size of the collection $\{R_{\beta,k}\}_{\beta \in B}$. If the randomized strategy must guess between an infinite collection of states, then there is no guarantee that the probability of selecting the correct preimage $R_{\beta,k}$ is non-zero. As an example, consider the problem in figure 4.3. In this example there is no horizontal position sensing, but there is perfect vertical position sensing and perfect velocity control. For the sake of example, let us assume that the system can only move vertically. The goal is a one-dimensional region specified by the slanted line. Clearly, the vertical lines drawn above the goal are all preimages, relative to a termination predicate that remembers the system's start region. [Similarly for vertical lines below the goal, of course.] This is because if the system knows on which vertical line it is located, then it knows at which height to stop a downward motion towards the goal. Now suppose that the system does not know its horizontal position, and thus consider a randomized strategy that decides to guess between the vertical lines. If the strategy guesses correctly, then the goal

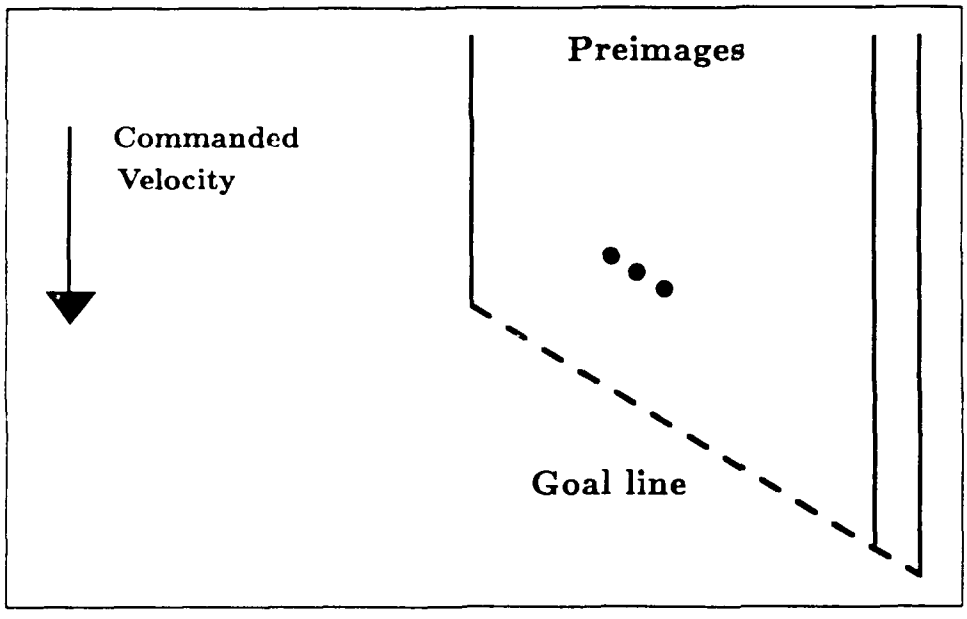


Figure 4.3: This example shows that preimages need not contain any interior, and that there may be an infinite number of preimages. In the example, vertical position sensing is perfect, horizontal position sensing is non-existent, and the system can only move vertically, with perfect velocity control. The goal is a line in space. Preimages are the vertical lines above the goal.

will be attained. However, the probability of guessing correctly is zero! In short, the randomized strategy is useless.[1]

The previous example makes two points: (1) That there may be an infinite number of preimages of a goal, and (2) that the probability of guessing the correct start state by guessing between an infinite number of preimages may be zero. However, the example was highly contrived, in that control uncertainty was taken to be zero and in that the sensing error was infinite in one dimension while zero in another. We will now explore some conditions that ensure a non-zero probability of guessing the correct starting state.

Constraints on Guessing Probabilities

Let us suppose that we have a collection of sets $\{R_\alpha\}$ that covers another set X. The set X describes the possible starting locations of the system. Each set R_α is a preimage. Here α is assumed to lie in some index set A. The operator SELECT chooses one of the sets R_α by selecting an α from A. Once an α has been selected, the system executes a strategy for attaining the goal from the preimage R_α, as outlined above and in chapter 3.

We assume that the choice of α is random. This means that we think of A as a measure space with some σ-algebra and some measure μ_A that determines how the α are chosen. Thus, the probability that α will lie in the set $B \subseteq A$ is given by $\mu_A(B)$. For instance, in the discrete case, A was just a finite subset of the integers $\{1, 2, \cdots, q\}$, and μ_A was given by $\mu_A(B) = |B|/|A|$ for every $B \subseteq A$.

Now, consider the actual state of the system $x \in X$. Let χ_{R_α} be the characteristic function of the set R_α. In other words, $\chi_{R_\alpha}(x)$ is 1 if $x \in R_\alpha$, and 0 otherwise. If one fixes x, and allows α to vary, then one can think of $\chi_{R_\alpha}(x)$ as a function of α. Thus the probability of correctly guessing a starting region R_α that contains the actual state of the system is given by:

$$p_c(x) = \int_A \chi_{R_\alpha}(x) d\mu_A(\alpha).$$

Said differently, $p_c(x) = \mu_A(B_x)$, where B_x is the set of all α for which $x \in R_\alpha$.

Now suppose that the state x is non-deterministically distributed over the region X. Thus an adversary could in principle choose x so as to minimize the probability of correctly guessing a region R_α. Thus, in choosing A and μ_A one must[2] satisfy the constraint

[1] We note in passing that this example did not satisfy the criterion of reliable goal recognition. However, it is easy to modify the example in the manner outlined in the section on the partial equivalence between sensorless and near-sensorless tasks (section 3.13.2), so as to achieve reliable goal recognition while preserving the character of the example.

[2] The term 'must' is bit stronger than necessary. With repeated guessing, it is fine if the probability of guessing correctly is zero occasionally, so long as the sum of the success probabilities over an infinite number of guesses is unity. However, constraint (4.2) is appropriate if one only considers individual guesses.

$$0 < \inf_{x \in X} p_c(x). \tag{4.2}$$

The constraint says that the probability of correctly guessing a preimage that includes the actual state of the system is non-zero, independent of the actual state of the system. Actually, the constraint says more, namely that the guessing probabilities are uniformly bounded away from zero. This ensures that the expected convergence time is finite in a loop that repeatedly guesses the start state.

Similarly, if the state x is randomly distributed over the set X, with probability measure $\nu(x)$, then one must satisfy the constraint

$$0 < \int_X p_c(x) d\nu(x). \tag{4.3}$$

This constraint says that the probability of guessing correctly is non-zero. The probability in this case is evaluated over both the state distribution and the guessing distribution.

Cautions

There are two cautions that should be mentioned with regard to repeated guessing attempts in the probabilistic case.

First, we must be careful in interpreting the probability integral of constraint (4.3). The probability $p_\nu = \int_X p_c(x) d\nu(x)$ is the probability of correctly guessing the starting state assuming that the state of the system is randomly distributed in accord with the distribution ν. This means that over a large number of *different problem instances* satisfying ν, the fraction of times that the system correctly guesses the starting state is given by p_ν. It does *not* necessarily mean that repeated guessing during execution of a single problem instance will yield a fraction of correct guesses that is roughly p_ν. This second interpretation is correct only if the distribution ν is created anew on each guessing loop, for instance, by purposefully executing a randomizing strategy that creates the distribution ν. The point is that the actual state of the system on a particular execution trial is some state x. If $p_c(x)$ is zero, then the system has zero probability of guessing correctly. Unless the system is made to change state appropriately between guesses, this probability will remain zero. Thus, even in the probabilistic case, it often makes sense to satisfy constraint (4.2) rather than merely constraint (4.3).

For instance, let us suppose that an incorrect guess always yields a strategy that does not affect the state of the system, while a correct guess yields a strategy that attains the goal. This is of course a strong assumption, but it will serve to illustrate our caution. Since the caution holds even in this special case, it certainly holds more generally. Ideally, we would hope that the probability of correctly guessing the starting state on the n^{th} guess is given by

$$(1 - p_\nu)^{n-1} p_\nu. \qquad \text{(incorrect)} \tag{4.4}$$

However, as we have said, this is an incorrect interpretation of p_ν. Instead, the probability of correctly guessing the starting state on the n^{th} guess is given by

$$\int_X (1 - p_c(x))^{n-1} p_c(x) d\nu(x). \qquad \text{(correct)} \tag{4.5}$$

Summing expression (4.4) over an infinite number of guesses yields unity. This need not be true for expression (4.5).

This brings us to our second caution. It is generally true that the measure ν varies on each guessing iteration. This is because each guess results in the execution of some strategy that affects the state of the system. This further complicates the description of success probabilities. Now the probability of correctly guessing the starting state on the n^{th} trial depends not only on the initial distribution ν, as in expression (4.5), but also on the previous guesses. We will not examine this issue in any detail.

Success Maximization

One could also postulate conditions on μ_A for maximizing the probability of successful goal attainment. This would involve considering the effect on the state of the system of executing a strategy derived from an incorrect guess. After all, in some cases an incorrect guess can still lead to goal attainment. We will not examine these conditions here.

Comparison of Non-Deterministic and Probabilistic Constraints

The difference between the non-deterministic constraint (4.2) and the probabilistic constraint (4.3) is the usual difference between a worst-case adversary and an average-case behavior. If we rewrite constraint (4.2) as

$$0 < \inf_{x \in X} \int_A \chi_{R_\alpha}(x) d\mu_A(\alpha) = \inf_{x \in X} \mu_A(B_x), \tag{4.6}$$

and if we rewrite constraint (4.3) as

$$0 < \int_{A \times X} \chi_{R_\alpha}(x) d(\mu_A \times \nu) = (\mu_A \times \nu)(D), \tag{4.7}$$

where $D = \{(\alpha, x) \mid x \in R_\alpha\}$, then this difference becomes clearer. In the non-deterministic case we want each of the slices B_x of D to have sufficient non-zero measure in the space A. In the probabilistic case we merely want the set D to have non-zero measure in the space $A \times X$. If we go back to the example of figure 4.3, and imagine that the starting state is uniformly distributed, then both A and X are essentially equal one-dimensional intervals, with the usual measures. The set D of successful guess-state pairs is the diagonal in the space $A \times X$, and hence of measure zero. If the goal were changed to a strip of finite width, then the vertical preimages would become non-degenerate rectangles, so that D would also become a strip of

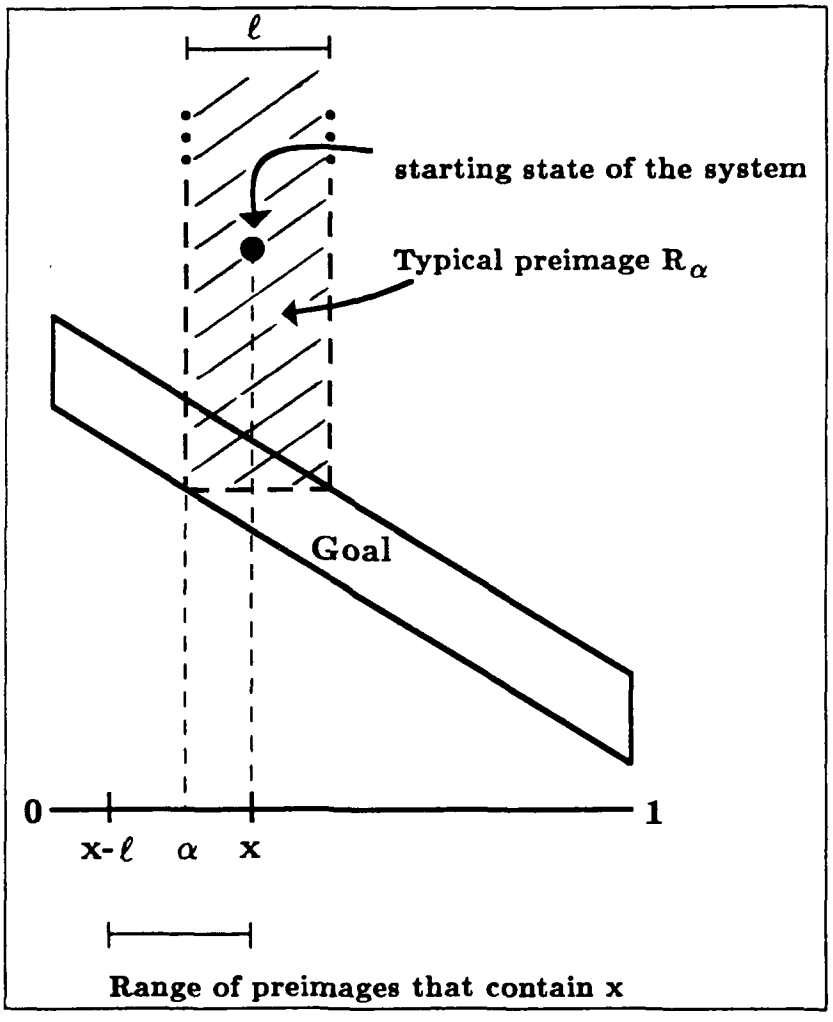


Figure 4.4: If the goal line of figure 4.3 is changed to a strip of non-zero width, then the preimages also become non-degenerate. This figure displays a typical such preimage. It is a vertical strip of width ℓ that contains the starting state of the system. See also figure 4.5.

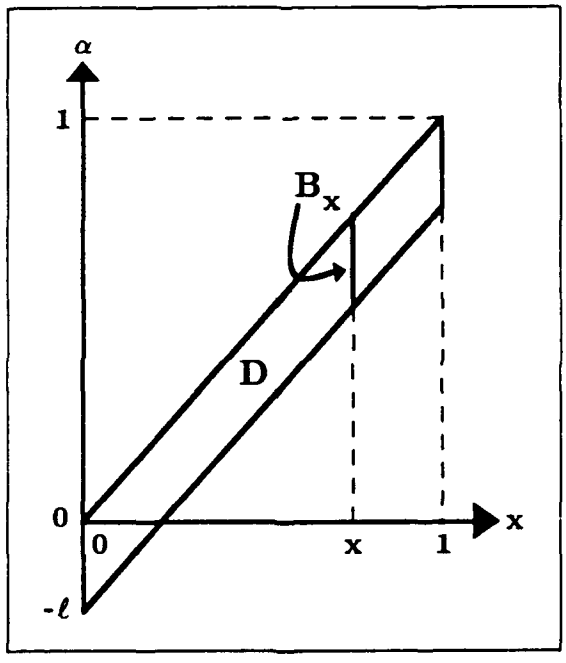


Figure 4.5: This figure graphs as a function of the system's x-coordinate the set of all preimages of figure 4.4 that contain the state of the system. B_x is the set of all preimages that contain x, and D is the union of these sets over all x. See also equations (4.6) and (4.7).

finite width. Suddenly the probability of success would be non-zero, despite there being an infinite number of preimages. See figures 4.4 and 4.5.[3]

4.2.2 Restricting SELECT to Finite Guesses

Let us focus on ensuring that the sets B_x each have non-zero measure in the non-deterministic setting, by satisfying (4.2), since satisfying this constraint automatically ensures that (4.3) holds as well. We will do this effectively by modifying the definition of SELECT and insisting that it only consider finite collections of covering preimages. In other words, the index set A is forced to be finite.

It is clear that this finiteness requirement imposes a fairly strong restriction on SELECT. In particular, the example of figure 4.3 does not satisfy the requirement. Nonetheless, in many instances the finiteness arises naturally. Consider for instance a set R that is covered by an infinite collection of sets $\{R_\beta\}$. Now suppose further that R is bounded, and that in fact the interiors of the R_β cover the closure of R, all in the usual topology on $\Re^n$. By compactness of the closure of R it thus follows that a finite subcollection of the R_β must actually cover R, as desired.

As stated so far, this explanation is not completely satisfactory. Among other things, the explanation does not properly take account of preimages that have no interior in $\Re^n$ because they lie on some surface of lower dimensionality. The main task remaining in this chapter is therefore to make more precise the naturality of the finiteness requirement. The explanation just given provides the basic outline of the argument.

Forward Projections

In order to motivate the insistence on coverage by interiors of preimages, we will consider the *forward projection* of a point moving with a commanded velocity that is subject to non-zero error. We defined the forward projection in the discrete setting on pages 94 and 102. In the continuous setting we need a time index as well, since actions are executed over some interval of time. Thus let us define *the forward projection at time t*, $F_{\mathbf{v}_0,t}(R)$, to be the set of configurations that the system might be in at time t, given that it started out in the region R at time zero, and moved during the time interval $[0,t]$ with commanded velocity $\mathbf{v}_0$, subject to control uncertainty as defined earlier. This notation differs slightly from that used in [Erd86]. If one is not interested in any particular time, then one may consider the *timeless forward projection*

[3]Of course, ignoring possible boundary effects, $p_c(x)$ is now non-zero for all x, since there is an interval of preimages that contain x, so in fact the guessing strategy would succeed even in the face of a worst-case adversary. Notice, however, that in order to avoid zero probabilities of success at the boundaries one has to almost unnaturally construct preimages that extend beyond the goal. Alternatively, one could only consider preimages R_α with α in the range $A = [0, 1-\ell]$, then insist that μ_A have positive measure on the atoms given by the endpoints 0 and $1-\ell$, as well as non-uniform measure near these endpoints. This is equivalent to constructing additional preimages of width less than ℓ near the endpoints.

$$F_{\mathbf{v}_0}(R) = \bigcup_{t \geq 0} F_{\mathbf{v}_0,t}(R).$$

For future reference, let us also recall the definition of a *backprojection* from [Erd86]. In particular, the backprojection $B_{\mathbf{v}_0}(G)$ of some region G is the set of all configurations from which the system is guaranteed to pass through the set G at some time, despite uncertainty, given that the commanded velocity is $\mathbf{v}_0$. Effectively, the forward projections encode the historical information available to the termination predicate, while the backprojections encode the reachability. See [Erd86] for further details.

For the sake of having a focused discussion, we will assume that the termination predicate is of the type discussed in the **LMT** work. In particular, in addition to the commanded velocity, the various uncertainty parameters, and a description of the environment, the termination predicate is given the following information: Initially, the termination predicate is given the start region. Thereafter, at every time $t \geq 0$, the termination predicate is given the current time, and the current sensory information. Of course, a particular termination predicate may only consider some of this information. The most powerful termination predicate, discussed in [Mas84], remembers all information given to it. It is assumed that the termination predicate can compute forward projections of any set for any time, as well as form arbitrary unions and intersections of these sets with themselves and with sensory interpretation sets.[4] A termination predicate signals goal attainment when its current knowledge state is inside a goal. For instance, consider a termination predicate that remembers the start region, and considers the current sensory information, but forgets past sensory information and does not look at the current time. If R is a preimage of the goals $\{G_\beta\}$ relative to a commanded velocity $\mathbf{v}_0$, this predicate will signal goal attainment when the set $F_{\mathbf{v}_0}(R) \bigcap B_{\epsilon_s}(\mathbf{x}^*)$ is inside some goal G_β. Here $\mathbf{x}^*$ is the current sensed position and $B_{\epsilon_s}(\mathbf{x}^*)$ is its interpretation set. More general descriptions exist for more general sensors.

Now[5] suppose that whenever velocity $\mathbf{v}_0$ is commanded the actual velocity lies in some error ball $B_\epsilon(\mathbf{v}_0)$. Note that ϵ may depend on $\mathbf{v}_0$. To avoid trivialities let us assume that $\epsilon < |\mathbf{v}_0|$. If $\mathbf{x} \in \Re^n$ lies in free space and t is non-zero but small enough so that the forward projection of $\mathbf{x}$ lies in free space, then we have that

$$F_{\mathbf{v}_0,t}(\{\mathbf{x}\}) = B_{t\epsilon}(\mathbf{x} + t\,\mathbf{v}_0).$$

In other words, for all non-zero times the forward projection of a point in free space is some open ball. Since $F_{\mathbf{v}_0,t}(R) = \bigcup_{\mathbf{x} \in R} F_{\mathbf{v}_0,t}(\{\mathbf{x}\})$, this says that for all non-zero times the forward projection of a set R in free space is an open set. Now consider the backprojection of some goal G, relative to a commanded velocity $\mathbf{v}_0$ and suppose that

[4]The term "compute" is used in a non-technical sense, that is, it is set-theoretic. Indeed, many sets are not computable in the technical sense of computability theory. See [CR] and [Can85] for some results on the computability and complexity of forward projections. See also [Erd84].

[5]Recall that $B_r(\mathbf{p})$ refers to the open ball of radius r about the point $\mathbf{p} \in \Re^n$.

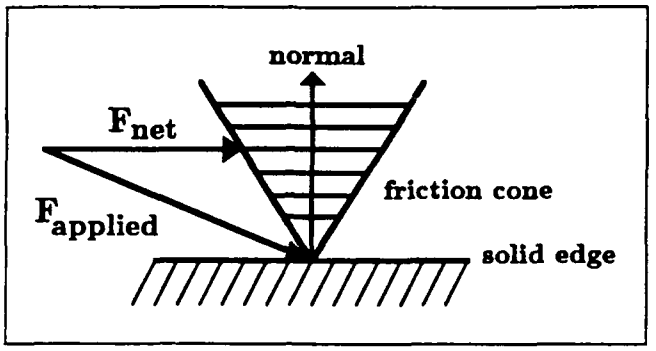


Figure 4.6: A two-dimensional friction cone. Also shown is the computation of a net force given an applied force.

this backprojection lies wholly in free space. By construction, any preimage R under velocity $\mathbf{v}_0$ of G must lie in this backprojection. Suppose that $R \subseteq \mathcal{B}_{\mathbf{v}_0}(G)$, and that t is a non-zero time at which the system cannot yet have encountered the goal. Then, even if R itself contains no interior, the set $F_{\mathbf{v}_0,t}(R)$ is open and all points in it are guaranteed to pass through the goal eventually. Of course, in general, the set $F_{\mathbf{v}_0,t}(R)$ need not be a preimage of G even if R is. However, for special cases, for instance if the termination predicate only uses the timeless forward projection and current sensed values in determining goal attainment, and if G is closed, then this forward projection is indeed a preimage of G (see [Erd86]). Thus there is strong motivation for considering only preimages with non-empty interior in guessing starting regions.

Collisions and Friction

In order to account for contact with obstacles, consider how the velocity error ball is modified by collisions with lower-dimensional surfaces in $\Re^n$. We will assume that all surfaces are piecewise linear, as is reasonable for polyhedral obstacles, and that friction is isotropic and invariant across any such planar patch. In particular, for hyperplanes of dimension $n-1$ friction is described by an n-dimensional cone with cone angle $\alpha = \arctan \mu$, where μ is the coefficient of friction.[6] The axis of this cone is the normal to the hyperplane. See figure 4.6 for the two-dimensional case in $\Re^2$. The effect of friction on the intersection of several such surfaces is determined by the generalized damper analogue of Newton's equations. In practice, we are thinking of $\Re^2$ and $\Re^3$. The description of friction in configuration spaces involving object rotations is slightly more complicated. In particular, the effective friction in configuration space may vary from configuration to configuration. Also, friction need not appear for certain tangential motions, such as those involving pure rotations. [See [Erd84].] For the case $\Re^2$, the computation of an effective motion is determined

[6] We will assume that static and sliding friction are equal.

by projecting the applied force onto the friction cone as indicated in figure 4.6. In particular, if the negative applied force $-\mathbf{F}_A$ lies inside the friction cone then there is no resulting motion. Otherwise, the net force is given by $\mathbf{F}_A + \mathbf{F}_R$, where $\mathbf{F}_R$ is a reaction force on the edge of the friction cone, whose normal component directly cancels the normal component of $\mathbf{F}_A$. For generalized damper dynamics, forces and velocities are equivalent, in the sense that the applied force is the term $\mathbf{B}\,\mathbf{v}_0$ in the equation $\mathbf{F} = \mathbf{B}\,(\mathbf{v} - \mathbf{v}_0)$.

More generally, if contact exists on some plane P of dimension $n-k$ in $\Re^n$, then given an applied force $\mathbf{F}_A$, an effective motion is computed as follows. One can think of the plane of dimension $n-k$ as being the intersection of k independent hyperplanes of dimension $n-1$. To say that the system is in contact with the plane P is to say that it is in contact with each of the individual hyperplanes. Let the outward unit normals to these hyperplanes be given by the vectors $\mathbf{n}_1, \ldots \mathbf{n}_k$.[7] Then the friction cone at the i^{th} point of contact is given by all forces that form an angle with the outward normal $\mathbf{n}_i$ that is no greater than $\alpha = \arctan \mu$. In other words, the friction cone is the set of forces

$$\mathcal{F}_i = \left\{ \mathbf{F} \;\middle|\; \mathbf{F} \cdot \mathbf{n}_i \geq \frac{|\mathbf{F}|}{\sqrt{1+\mu^2}} \right\}.$$

Said differently, the set of reaction forces that can be generated by the i^{th} hyperplane is given by the set $\mathcal{F}_i$. The composite friction cone due to contact with the plane P is simply the vector sum of the individual friction cones. In other words, the set of possible reaction forces is the set $\mathcal{F}_P$, where

$$\mathcal{F}_P = \left\{ \mathbf{F} \;\middle|\; \mathbf{F} = \sum_{i=1}^{k} \mathbf{F}_i, \text{ with } \mathbf{F}_i \in \mathcal{F}_i \right\}.$$

Given an applied force $\mathbf{F}_A$, there are two possibilities. Either the system moves or it sticks. Consider the case in which it sticks. Then we must have that the reaction force $\mathbf{F}_R$ is of the form $\mathbf{F}_R = -\mathbf{F}_A$. In other words, $-\mathbf{F}_A \in \mathcal{F}_P$. Conversely, if $-\mathbf{F}_A \in \mathcal{F}_P$, then a possible motion solution is given by sticking, with reaction force $\mathbf{F}_R = -\mathbf{F}_A$.

We will consider the other possibility, in which the system moves, presently. First, let us observe that in general the effective contact may not involve actual contact with all the hyperplanes that define the plane P. For instance, if the applied force points away from each of these planes, that is, if $\mathbf{F}_A \cdot \mathbf{n}_i > 0$, for all $i = 1, \ldots, k$, then the system is effectively not in contact with any of the hyperplanes. Thus a possible reaction force would be $\mathbf{F}_R = \mathbf{0}$, meaning that the motion of the system would be through free space, along the direction specified by $\mathbf{F}_A$.

In general, any subset of the k hyperplanes defining the contact with P might actually constitute the effective contact. Any such smaller contact set defines a higher-dimensional plane of contact. In principle one therefore needs to recursively check all

[7] In general, we should also allow redundant constraints and perhaps different coefficients of friction on the different hypersurfaces. The discussion may be generalized to include these.

these smaller contact sets for possible reaction forces. For each of these one computes the net resulting motion, in the manner to be outlined presently. If such a net motion is consistent with all the kinematic constraints, then it is a feasible solution to the motion problem.

In the most general case, there may be several solutions consistent with the applied force F_A. This is particularly true when in contact with low-dimensional surfaces. The resulting motion may depend on one's interpretation of these contacts. For instance, in two dimensions, contact with a convex vertex may be thought of in one of three ways: contact with the edge on one side of the vertex, contact with the edge on the other side of the vertex, or simultaneous contact with both edges. Thus several possible contact states may be consistent with Newton's equations and Coulomb friction (or their analogues under generalized damper dynamics). This ambiguity may introduce further non-determinism into the system's behavior.

Let us assume that the effective contact is given by all the k hyperplanes that define the plane P. Now consider the possibility that the system moves. We can write the applied force as $\mathbf{F}_A = \mathbf{F}_n + \mathbf{F}_t$, where $\mathbf{F}_n$ lies in the normal space spanned by the k normals, and $\mathbf{F}_t$ is parallel to the plane P. Similarly, since contact with P is maintained, observe that the reaction force is of the form $\mathbf{F}_R = -\mathbf{F}_n - g\,\hat{\mathbf{t}}$, where $g \geq 0$ and $\hat{\mathbf{t}}$ is some unit vector parallel to the plane P. We will see shortly that $\hat{\mathbf{t}}$ is positively parallel to the tangential component $\mathbf{F}_t$ of the applied force. In any event, the net force is of the form $\mathbf{F}_{\text{NET}} = \mathbf{F}_t - g\,\hat{\mathbf{t}}$. If this vector is non-zero, then it specifies the direction of motion, in the plane P.

Since the system is moving, and in contact with each of the hyperplanes, the isotropy assumption implies that the reaction force at each of the contributing hyperplanes must lie on the edge of its respective friction cone.[8] Furthermore, each reaction force must oppose the direction of motion. This says that the tangential component of the reaction force at each of the k hyperplanes is actually anti-parallel to the vector $\mathbf{F}_{\text{NET}}$. In turn, this means that $\hat{\mathbf{t}}$ is positively parallel to $\mathbf{F}_{\text{NET}}$, and hence to $\mathbf{F}_t$. We see therefore that the frictional part of the reaction forces does not contribute to the maintenance of contact with the plane P, but merely to the reduction of tangential motion. By construction we have that $\mathbf{F}_n = c_1\,\mathbf{n}_1 + \cdots + c_k\,\mathbf{n}_k$, for some set of constants $\{c_i\}$. It follows that each of the c_i in the description of $\mathbf{F}_A$ must be zero or negative, for otherwise the normal reaction forces at the points of contact would be negative, a physical impossibility. The scalar g thus may be written as the sum $g_1 + \cdots + g_k$, where friction dictates that $0 \leq g_i \leq -\mu c_i$, for all i.

In short, for contact with the plane P under applied force $\mathbf{F}_A$, there are two possibilities that do not involve the breaking of contact. First, the negative applied force may lie inside the composite friction cone $\mathcal{F}_P$, in which case the resulting motion may be zero. Of course, under certain indeterminacies a tangential motion may be possible as well, for instance, if we permit a set of dependent normals, that is, a

[8] This need not be true in general configuration spaces, such as those involving rotations, or multiple moving objects that do not interact. In those spaces the isotropy assumption does not hold. Generalizations of the procedure for computing net motions apply, although the specific conclusions need not.

set of redundant hyperplanes, along with different coefficients of friction on each of the hyperplanes. Second, if the negative applied force lies outside of the composite friction cone, and contact is not broken, then a tangential motion must result. If a tangential motion does occur, then the tangential reaction force has magnitude $g = -\mu(c_1 + \cdots c_k)$, with $c_i \leq 0$ for all i. The resulting motion is determined by the net force $(|\mathbf{F}_t| - g)\,\hat{\mathbf{t}}$, which points in the same tangential direction as the applied force, but has a smaller magnitude. [Note that this only makes sense if $g < |\mathbf{F}_t|$.]

Forward Projections on Surfaces

The discussion of the last few paragraphs is intended partly as a quick review of friction. The main purpose, however, is to indicate that the forward projection of a point on a surface, by those velocities in the velocity error ball that maintain contact with the surface, forms a set that is open in the relative topology of the surface. In other words, suppose $\mathbf{x}$ is some point on a plane P of dimension $n - k$ as above. Consider applying a nominal commanded velocity $\mathbf{v}_0$ subject to the usual uncertainty considerations. There are two possibilities. Either the point moves away from the surface, or it maintains contact. Of course, in some cases, over time the point may be able to do both, and intermittently hop back and forth between free space and contact space, and perhaps between surfaces of different dimensionality.

Suppose the commanded velocity is $\mathbf{v}_0$ and that all effective commanded velocities lie in the open ball $B_\epsilon(\mathbf{v}_0)$. Suppose further that the system is in contact with a plane P of dimension $n - k$, formed by the intersection of k hyperplanes. Let the independent unit normals of the defining k hyperplanes be given by $\mathbf{n}_1, \ldots, \mathbf{n}_k$. Then any vector can be written as $\mathbf{v} = \sum_{i=1}^{k} c_i \mathbf{n}_i + h\,\hat{\mathbf{t}}$, where the $\{c_i\}$ and h are scalars and $\hat{\mathbf{t}}$ is some unit tangent vector parallel to the plane P. Assuming generalized damper dynamics with an identity damping matrix $\mathbf{B}$, we will think of forces and velocities as equivalent. The set of velocities in $B_\epsilon(\mathbf{v}_0)$ that can maintain contact with the plane P is given by

$$B_{\text{contact}} = \left\{ \mathbf{v} \in B_\epsilon(\mathbf{v}_0) \;\middle|\; \mathbf{v} = \sum_{i=1}^{k} c_i \mathbf{n}_i + h\,\hat{\mathbf{t}}, \text{for some } h \text{ and } \hat{\mathbf{t}}, \text{ with } c_i \leq 0 \text{ for all } i \right\}$$

$$\bigcup \left\{ \mathbf{v} \in B_\epsilon(\mathbf{v}_0) \;\middle|\; -\mathbf{v} \in \mathcal{F}_P \right\}.$$

The set of all velocities that must break contact is given by $B_{\text{break}} = B_\epsilon(\mathbf{v}_0) - B_{\text{contact}}$. By breaking contact we mean simply that the instantaneous contact may be thought to occur either in free space or on a plane of higher dimension. Contact state may also be changed by other means, say by sliding off the boundary of the plane and into free space. That may be viewed as a change of state, in that the system is in contact at some time $t = t_0$, but is in free space at time $t > t_0$.

Finally, the set of velocities that can break contact completely is given by

$$B_{\text{free}} = \{\mathbf{v} \in B_\epsilon(\mathbf{v}_0) \mid \mathbf{v} \cdot \mathbf{n}_i > 0, i = 1, \ldots, k\}.$$

These are all velocities for which the system could move into free space.

The set B_{contact} is relatively closed in the ball $B_\epsilon(\mathbf{v}_0)$, so the set B_{break} is open. More to the point, we see that the set of velocities that can break contact completely, that is, the set B_{free}, is an open set. Thus, by an argument similar to the one given above, the portion of the forward projection that arises solely from velocities that move through free space is an open subset of free space.

Let us focus now on the contact with the plane P, and show that the forward projection of a point on this plane, by velocities in the error ball that can maintain contact with the plane, contains an interior in the relative topology of the plane. Given an applied force or velocity $\mathbf{v}$, let us denote by $\pi(\mathbf{v})$ the resulting net force or velocity, assuming generalized damper dynamics and contact with P. For some $\mathbf{v}$, $\pi(\mathbf{v})$ will just be zero, that is, no motion will result. We would like to show that the set of net velocities, given by $\pi(B_{\text{contact}})$ is a set with interior (relative to the topology of $\Re^{n-k}$). In fact, we will show that almost all resulting velocities are interior to the set $\pi(B_{\text{contact}})$. The only exception will be in some cases the zero velocity. This implies that if one only considers non-sticking contact velocities, then the forward projection of any point will be an open set in the relative topology of the contact plane. The argument is the same as for the free space case except that now the velocity error ball is replaced by some other open set of possible velocities. If there are sticking velocities in the projected velocity error ball then the forward projection of a region R will be the union of some relatively open set determined by the non sticking velocities and the region R itself. Thus the forward projection contains an interior. More importantly, if one is only interested in preimages, then there can be no sticking velocities, as otherwise one could not guarantee goal attainment, so the forward projection with respect to contact velocities is a relatively open set.

Let us therefore consider those velocities for which the resulting motion is not zero. By the discussion above, we can write the effect of π on such a velocity as:

$$\pi\Big(\sum_{i=1}^{k} c_i \mathbf{n}_i + h\,\hat{\mathbf{t}}\Big) = \Big(h + \mu \sum_{i=1}^{k} c_i\Big)\,\hat{\mathbf{t}}.$$

Now suppose $\mathbf{v} = h\,\hat{\mathbf{t}}$, that is, suppose all of the c_i are zero. Then $\pi(\mathbf{v}) = \mathbf{v}$, that is, π restricted to velocities with no normal component is just the identity map. More generally, if one fixes the constants $\{c_i\}$ at some set of values, then one can think of π as a self-mapping between tangent vectors in the plane of contact. The mapping is a form of shifting given by $\pi_{\{c_i\}}(h\,\hat{\mathbf{t}}) = (h + c)\,\hat{\mathbf{t}}$, with $c = \mu \sum_{i=1}^{k} c_i$, and $h > 0$. Clearly $\pi_{\{c_i\}}$ is well-defined for all non-zero tangent vectors. However, the assumption that the applied velocity results in motion means that we are only applying π to vectors for which $h > -c \geq 0$. Let us define two sets of tangent vectors in the plane of contact. Let V_0 be the set of all tangent vectors which can be written in the form $h\,\hat{\mathbf{t}}$ with $h > 0$, and let V_c be the set of all tangent vectors $h\,\hat{\mathbf{t}}$ with $h > -c$. Then $\pi_{\{c_i\}}$ is a one to one mapping of V_c onto V_0. Thus $\pi_{\{c_i\}}$ possesses a two-sided inverse,

mapping V_0 onto V_c. The inverse is given by $\pi^{-1}_{\{c_i\}}(\ell\,\hat{\mathbf{t}}) = (\ell - c)\,\hat{\mathbf{t}}$, for all unit tangent vectors $\hat{\mathbf{t}}$ and scalars $\ell > 0$. It is clear that this inverse is a continuous function, and so we see that $\pi_{\{c_i\}}$ is an open map from V_c to V_0.

Now fix a particular non-zero image vector of π applied to B_{contact}. This vector is of the form $\pi(\mathbf{v})$, for some vector $\mathbf{v} = \sum_{i=1}^{k} c_i\,\mathbf{n}_i + h\,\hat{\mathbf{t}}$ in the set B_{contact}, with $h > -c = -\sum_{i=1}^{k} c_i \geq 0$. Consider the set of velocities

$$\mathcal{O} = \left\{ \mathbf{w} \in B_{\text{contact}} \;\middle|\; \mathbf{w} = \sum_{i=1}^{k} d_i\,\mathbf{n}_i + \mathbf{v}_t, \text{ with } |d_i - c_i| < \delta \text{ and } |\mathbf{v}_t - h\,\hat{\mathbf{t}}| < \tfrac{h+c}{2}\,\delta \right\}.$$

Here δ is some small positive number, and $\mathbf{v}_t$ is any tangent vector parallel to the contact plane P. The set $\mathcal{O}$ is an open neighborhood in B_{contact} of the velocity $\mathbf{v}$. If δ is chosen small enough then one can guarantee that all the vectors in the right part of the set definition actually lie inside the velocity error ball $B_\epsilon(\mathbf{v}_0)$, since it is an open ball. This says that $\mathcal{O}$ wholly contains the set $\mathcal{O}_c = \left\{ \mathbf{w} \;\middle|\; \mathbf{w} = \sum_{i=1}^{k} c_i\,\mathbf{n}_i + \mathbf{v}_t, \text{ with } |\mathbf{v}_t - h\,\hat{\mathbf{t}}| < \tfrac{h+c}{2}\,\delta \right\}$. Note that in this last set the normal components are all the same, determined by the $\{c_i\}$ that define $\mathbf{v}$. Viewing this set as a subset of the tangent vectors to the contact plane P, we see that it is relatively open and a subset of V_c. But this says that the image $\pi_{\{c_i\}}(\mathcal{O}_c)$ is an open neighborhood of $\pi(\mathbf{v})$, and thus $\pi(\mathcal{O}_c)$ is a neighborhood of $\pi(\mathbf{v})$. This shows that $\pi(B_{\text{contact}}) - \{\mathbf{0}\}$ is an open set in the topology of $\Re^{n-k}$.

Contact Changes

Finally, suppose that we model obstacles as closed sets. Additionally, each plane of any dimension is a closed set. Now consider the possible contact changes for a portion of the forward projection that is an open set relative to its contact state. For instance, consider an open ball of dimension $n-k$ on a plane of dimension $n-k$, and consider its collision with a subplane of dimension $n-k-i$. The intersection with the lower-dimensional plane is necessarily relatively open in that plane. Conversely, suppose that at time $t = t_0$ an open ball of dimension $n-k-i$ prepares to lift off from from the subplane of dimension $n-k-i$, moving off into the containing plane of dimension $n-k$ for all times $t > t_0$. Given only velocities for which it is possible to move on the plane of dimension $n-k$, the arguments above show that this ball forward projects into an open set in the relative topology of the containing plane for all times $t > t_0$.

Brief Summary

In short, we have shown that the forward projection of a set relative to an open velocity uncertainty ball contains interior relative to each of the contact states it defines. The argument above is not a formal proof, but it does provide some intuition and some motivation for insisting that the operator SELECT only guess between finite collections of preimages.

Compactness Argument

We are now in a position to state the compactness argument more generally. First, let us write the reachable state space X in the form

$$X = K_n \bigcup \cdots \bigcup K_1 \bigcup K_0,$$

where K_i is the closure of the set of all points that lie on a plane of dimension i. This means that K_n is the closure of the set of all points in X that lie in free space, while K_0 is the set of all vertices of the obstacle polyhedra. We assume that all polyhedra have full dimensionality, that is, are formed by sets of hyperplanes of dimension $n-1$. Then we see that $K_n \supset \cdots \supset K_1 \supset K_0$.

If we are given a region $R \subseteq X$, we can thus form the unique union $R = R_n \bigcup \cdots \bigcup R_0$, where $R_i = R \bigcap K_i$. Similarly, given a collection of preimages $\{R_\beta\}$ that cover R, we can form the collections $\{R_{\beta,n}\}, \ldots \{R_{\beta,0}\}$, where $R_{\beta,i} = R_\beta \bigcap K_i$, for all β and i. Notice that each $R_{\beta,i}$ is a preimage since the subset of a preimage is always also a preimage. Clearly, each collection $\{R_{\beta,i}\}$ covers the dimensionally corresponding subset R_i of R. If we assume that the set R is compact to begin with, and that the preimages $\{R_{\beta,i}\}$ are open in the relative topology of K_i, then in fact a finite number of these preimages will cover R_i. Thus SELECT can naturally choose between a finite set of preimages. Actually, we can further loosen the openness requirement on the preimages, and merely ask that each of the sets $R_i \bigcap R_{\beta,i}$ be open in the relative topology of R_i. This permits the preimages to contain some extra limit points.[9]

Preimages and Forward Projections

We have tried to motivate the discussion of open preimages or preimages with interior by showing that the forward projection naturally contains interior in each dimension, for tasks in $\Re^n$ that involve polyhedral obstacles with simple friction, and that use generalized damper dynamics subject to non-vanishing control uncertainty. If one therefore insists that preimages at least contain their forward projections for a small period of time, then one can guarantee that preimages contain interior as well.

Clearly there will still be some problems for which infinite coverage by preimages without interior is unavoidable. In such cases, if the unrestricted version of SELECT is to function properly, one must satisfy one of the conditions (4.2) or (4.3). In general, this will entail looking at each particular guessing step individually, then determining an appropriate index set A and guessing distribution μ_A. However, for many problems one may restrict SELECT to finite decisions.

Let us briefly also indicate why it is reasonable to insist that preimages contain part of their forward projection. For special cases, as we have noted, it is almost automatic. More generally, the argument is very similar to the one used to establish the openness of the forward projection. Consider a preimage R and its forward

[9] For instance the semi-open interval $[0, \frac{1}{2})$ is open in the relative topology of the closed interval $[0, 1]$.

projection $F_{\mathbf{v}_0,t}(R)$ at some time $t > 0$. We claim that if t is small enough, then a relatively open subset of this forward projection is itself a preimage. This does not quite say that R contains any interior, but it does say that there is a preimage naturally derivable from R that does contain interior (in fact is open). In order to make the argument we must make two further assumptions: (1) That the sensing uncertainty is a non-degenerate open ball, and (2) that there is some minimum time $t_{\min} > 0$ before which goal attainment and recognition is impossible from the preimage R. One can probably remove this second assumption, but we will not worry about that here. Additionally, we will focus on the case in which the forward projection lies in free space and in which the only sensor is a position sensor.

In order to be concrete, let us suppose that R is a preimage of some collection of goals $\{G_\beta\}$, relative to the commanded velocity $\mathbf{v}_0$ and some termination predicate. Suppose that the control uncertainty ball has radius $\epsilon = \epsilon(\mathbf{v}_0)$, while the position sensing uncertainty ball has radius ϵ_s. Choose $t_0 > 0$ to be smaller than both $t_{\min}$ and $\frac{1}{2}\frac{\epsilon_s}{|\mathbf{v}_0|+\epsilon}$. In other words, in the time t_0, the furthest any point can move is half the radius of the sensing uncertainty ball. Now consider any subset R_0 of R whose diameter is less than $\epsilon_s/2$. Let $F = F_{\mathbf{v}_0,t_0}(R_0)$, which is a relatively open subset of $F_{\mathbf{v}_0,t_0}(R)$. We would like to establish that F is a preimage of the collection $\{G_\beta\}$, relative to the commanded velocity $\mathbf{v}_0$ and the same termination predicate as that used for the preimage R.

In order to establish that F is a preimage we must show that any trajectory starting in this set is guaranteed to terminate recognizably inside a goal. First, let us note that all trajectories emanating from F must pass through a goal by the definition of R and $t_{\min}$. Next,[10] consider a motion starting in F at time $t' = 0$. Let us determine the information available to the termination predicate at time $t' = 0$. In line with the discussion on page 209, the termination predicate is given the start region F, the time $t' = 0$, and whatever sensed value $\mathbf{x}^*$ is returned by the sensor at time $t' = 0$. In general, $\mathbf{x}^*$ can be any sensory value consistent with a starting position in F. Of course, it is possible that the particular termination predicate employed will ignore some or all of this information. Let us denote by $K_{\mathbf{x}^*}$ the knowledge state derived by the termination predicate from the information it is given at time $t' = 0$.

Since R is a preimage, so is R_0. Consider therefore a motion emanating from the set R_0. Fix some $\mathbf{x}_0 \in R_0$. Given an adversarial sensor, we may assume that the sensory value returned for all times t in the range $[0, t_0)$ is $\mathbf{x}_0$. By construction, the forward projection of R_0 at any such time is contained inside the sensing error ball about $\mathbf{x}_0$, that is, $F_{\mathbf{v}_0,t}(R_0) \subseteq B_{\epsilon_s}(\mathbf{x}_0)$ for all $t \in [0, t_0]$. In short, the sensors contribute nothing over that time interval to the termination predicate's decision. Thus the knowledge state K available to the termination predicate at time $t = t_0$ (before sensing) is some superset of F, and possibly equal to F. Since the motion starting from R_0 at time $t = 0$ is guaranteed to terminate recognizably in a goal, the motion starting from F at time $t = t_0$ with knowledge state K and sensor value $\mathbf{x}^*$

[10] We use the notation t' to indicate that this time is not directly related to the clock used in executing a motion from the preimage R.

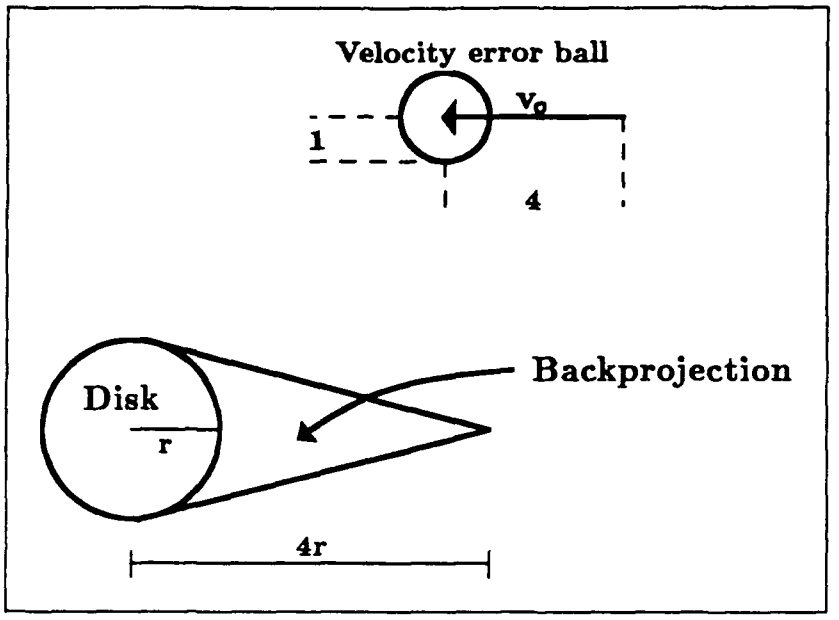


Figure 4.7: This figure shows a typical backprojection of a two-dimensional disk.

must terminate recognizably in a goal. Denote by K^* the knowledge state formed from K and x^*. Now consider again the termination predicate that starts a motion in the set F at time $t' = 0$. Recall that the knowledge state available to this termination predicate is $K_{\mathbf{x}^*}$. It is reasonable to assume that the knowledge state K^* is a superset of the knowledge state $K_{\mathbf{x}^*}$, which establishes that F is a preimage. [11]

One observes that a similar argument shows that the set

$$\bigcup_{0<t\leq t_0} F_{\mathbf{v}_0,t}(R_0)$$

is also a preimage that is an open subset of the forward projection of R, assuming that the termination predicate does not consider time. More general forward projections are preimages if the termination predicate only considers current sensed values.

Finite Guesses

A final comment should be made. We have thus far indicated the existence of preimages with interior. We have not yet motivated the naturality of insisting that open preimages cover a compact guessing region. This condition is desirable since it ensures guessing finiteness. However, there are many cases in which the guessing

[11] One can imagine termination predicates that randomly choose starting knowledge states that include the actual starting region, but we will exclude those. Most likely K^* and $K_{\mathbf{x}^*}$ are actually equal. This is certainly true for most of the variations of termination predicates discussed in [Erd86].

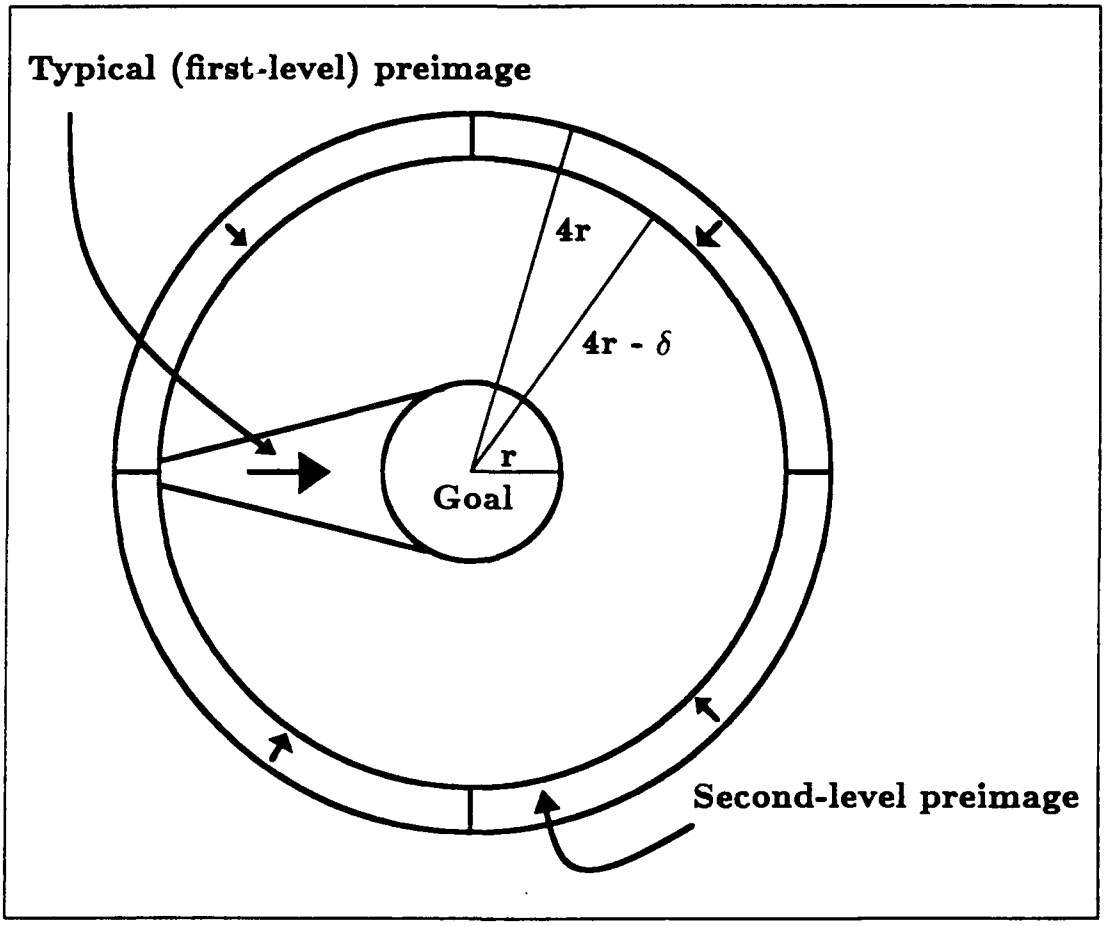


Figure 4.8: The union of all possible backprojections of the disk of figure 4.7 is a disk of radius $4r$. This figure shows how to split the disk into a finite number of regions. A guessing strategy that cannot sense the position of the system can thus guess the correct region with non-zero probability. If the system guesses that it is in the outer ring of width δ, then it moves inward as indicated by the arrows. Otherwise, the system guesses that it is in some backprojection, one of which is shown.

region is open rather than closed. We show now, by example, that this poses no serious problem.

Consider the task of attaining an open disk in the plane. Assume that the velocity uncertainty is given by a ball with radius $\epsilon = \frac{1}{4}|v_0|$, where v_0 is the commanded velocity, as usual. If the disk has radius r, this says that one can backproject along any direction, and obtain a cone with apex at distance $4r$ from the center of the disk. See figure 4.7. Suppose that the disk is recognizable once entered. Then each of these backprojections is actually a preimage, relative to a termination predicate that only checks for disk attainment. This says that there is a collection of preimages whose interiors cover the open ball B_{4r} of radius $4r$ centered at the center of the disk. However, the interiors of the preimages do not cover the closed ball of radius $4r$. Thus, if the initial position is known to lie in B_{4r} the finite version of SELECT does not apply, that is, there is no finite collection of preimages that covers B_{4r}. We note in passing that this need not be a problem if the starting position is probabilistically distributed and constraint (4.3) holds. In other words, for some probabilistic distributions, SELECT can successfully choose between the infinite collection of preimages, by, for instance, guessing the angle of approach. However, in the non-deterministic setting, constraint (4.2) does not hold, and one really does need some finite version of SELECT. To see that this is possible, imagine shrinking the ball B_{4r} slightly, so that it only has radius $4r - \delta$, with $0 < \delta < r$. Now a finite number of preimages covers this ball $B_{4r-\delta}$. Thus, whenever the actual position lies in $B_{4r-\delta}$, the probability that SELECT will guess the correct preimage is uniformly bounded away from zero. Furthermore, one can split the annulus $B_{4r} - B_{4r-\delta}$ into a finite number of regions, each of which is preimage of the ball $B_{4r-\delta}$ for some velocity and a termination predicate that keeps track of time. Thus one can change the problem into a multi-step multi-guess randomization, and ensure that constraint (4.2) is satisfied. See figure 4.8. This approach applies more generally.

4.3 Summary

This chapter explored in the continuous domain the analogue to the randomized strategies developed in chapter 3 for the discrete domain. The chapter first reviewed the **LMT** preimage methodology for planning guaranteed strategies in the presence of uncertainty. This framework was used as the basis for defining randomized strategies, much as dynamic programming was used in the discrete domain. One of the difficulties in the continuous case is the need for randomizing between a possibly infinite number of decisions. The chapter exhibited conditions under which infinite decisions still yield non-zero probabilities of success. Further, it was shown that in many cases apparently infinite decisions may be reduced to a finite number of choices.

Chapter 5

Diffusions and Simple Feedback Loops

In this chapter we will explore the continuous version of discrete random walks. In continuum spaces the natural analogue to a random walk is a diffusion process.

We saw in the discrete setting that random walks on graphs constitute a simple type of randomized strategy, in which the future behavior of the system depends probabilistically only on the current state and not on any past states. Simple feedback loops constitute an important class of random walks, whenever the control and sensing errors are probabilistically distributed. Recall that in a simple feedback loop the current action to be executed is determined solely as a function of current sensory information, without any reference to previous sensory values.

We already caught a glimpse of the behavior of a continuous simple feedback loop in the introductory example of section 2.4. In that example we assumed an error distribution consisting of a fixed bias. More generally, we would like to have a language for describing the behavior of the strategy of that example for various distributions. In particular, we would like to determine the convergence times of the strategy for certain simple common error distributions, such as unbiased Gaussians. Fast specialized strategies are known in these cases. The randomized strategy is formulated to succeed independent of the actual error distributions, so long as these distributions satisfy certain bounds. However, the speed of convergence of the strategy depends on the actual error distributions. If the speed of convergence is reasonably quick in the simple settings then it makes sense to employ the generally applicable randomized strategy rather than to seek and employ a specialized fast strategy for each possible instantiation of error distributions.

In the discrete setting the notions of progress measure and expected progress provided a convenient tool for discussing the behavior and convergence times of random walks. These same notions carry over to the continuous setting. Indeed the notion of an expected local velocity arises in the very definition of a diffusion process.

This chapter will briefly review some basic facts from diffusion theory, then turn to examples. We will not restate or reprove all the results from discrete random walks in

the continuous setting. Instead, the main aim of this chapter is to develop an approach for analyzing simple feedback strategies of the type discussed in section 2.4. These strategies execute actions designed to make progress along some progress measure whenever the current sensory information permits such progress, and otherwise they execute a randomizing action. The randomizing actions ensure that the system will not become stuck hopelessly in some region from which progress is impossible. We will focus in particular on a fairly detailed analysis of the randomized strategy for attaining a two-dimensional hole, as in the example of section 2.4.

5.1 Diffusions

A diffusion process is basically the continuous version of a random walk. The important quantities that govern the behavior of a diffusion process are the local *drift* and *variance* at each point in the state space. These measure the expected velocity at each point and the variance in that expectation. Additionally, diffusion processes satisfy a continuity requirement, which ensures that nearly all sample paths of the process are continuous. This requirement therefore excludes processes that make random jumps, such as the first randomized strategy suggested for the example of section 2.4. Other processes excluded are those in which history plays a role in determining the future behavior of the system. In other words, diffusion processes must be Markovian.

The following material is a condensed version of the discussion of diffusion processes found in [KT2] and [FellerII].

First, let us assume that the state space of our diffusion process is $\Re^n$, and let us denote the process by $\{\mathbf{X}(t), t \geq 0\}$. In other words, $\mathbf{X}(t) \in \Re^n$ is a random variable describing the state of the system at time t. The Markovian nature of the process means that there exists a function $Q_{t,\Delta t}(\mathbf{x}, \mathbf{y})$, which describes the probabilistic transition function of the process over the time interval $[t, t + \Delta t]$. This function plays the role of the probability matrix (p_{ij}) in the discrete case. In particular, if the system is in state $\mathbf{x}$ at time t, then the probability that it will be in a state in the set Y at time $t + \Delta t$ is given by

$$\int_Y Q_{t,\Delta t}(\mathbf{x}, \mathbf{y}) d\mathbf{y}.$$

The continuity condition then takes the form,

$$\lim_{\Delta t \downarrow 0} \frac{1}{\Delta t} \int_{|\mathbf{y}-\mathbf{x}| \geq \delta} Q_{t,\Delta t}(\mathbf{x}, \mathbf{y}) d\mathbf{y} = 0,$$

for all positive δ.

In keeping with standard notation, we will use the symbol "$E[\,\cdot\,]$" to denote the expectation of whatever "$\cdot$" represents. The probability space for computing this expectation will generally be clear from context. Following [KT2], we will let $\Delta_h \mathbf{X}(t)$ be the change in the process over time interval h, that is, $\Delta_h \mathbf{X}(t) = \mathbf{X}(t + h) - \mathbf{X}(t)$.

The *local* or *infinitesimal drift* $\mu(\mathbf{x},t)$ and *variance* $\sigma^2(\mathbf{x},t)$ are given by the formulas: [1]

$$\mu(\mathbf{x},t) = \lim_{h \downarrow 0} \frac{1}{h} E[\,\Delta_h \mathbf{X}(t) \,|\, \mathbf{X}(t) = \mathbf{x}\,], \tag{5.1}$$

$$\sigma^2(\mathbf{x},t) = \lim_{h \downarrow 0} \frac{1}{h} E[\,\{\Delta_h \mathbf{X}(t)\}^2 \,|\, \mathbf{X}(t) = \mathbf{x}\,]. \tag{5.2}$$

Here μ is a vector of dimension n that represents the expected velocity of the process, while σ^2 and $\{\Delta_h \mathbf{X}(t)\}^2$ are matrices of dimension $n \times n$ that essentially measure the autocorrelation of the process. We will also refer to local drift as *expected infinitesimal velocity* and as *expected velocity*, indicating that this notion of velocity is a probabilistic average over possible displacements.

As pointed out in [KT2], under certain regularity conditions, a Markov process is known to be a diffusion process if the following condition holds for some $p > 2$. The limit is assumed to exist uniformly in $\mathbf{x}$ over any compact subset of the state space.

$$\lim_{h \downarrow 0} \frac{1}{h} E[\,|\Delta_h \mathbf{X}(t)|^p \,|\, \mathbf{X}(t) = \mathbf{x}\,] = 0. \tag{5.3}$$

5.1.1 Convergence to Diffusions

For the applications in which we are interested the resulting strategies are not diffusion processes. This is because sensing and action generally occur at discrete time intervals, rather than continuously, so that the process is not strictly speaking Markovian at each location and instant of time. A correct description of these strategies would therefore model each as a sequence of actions executed in a continuous space at discrete, not necessarily regularly spaced, time intervals. For each action one would define a probability transition kernel Q as above, then chain several such actions together by convolving these kernels in the manner outlined by the Chapman-Kolmogorov equation.[2] However, this approach obscures some of the basic issues that are of concern to us, namely whether the process is making progress towards the goal, and if so, how fast it is moving. Fortunately, many discrete time processes may be thought of as part of a sequence of such processes that converges to a diffusion process. In such cases the diffusion process may well approximate the discrete-time process. In these cases, an analysis of the diffusion process provides as well an approximate analysis of the discrete-time process. We will not worry about the details of such approximations, but simply point to [KT2] for a brief introduction. In this chapter we will assume that our discrete representations may be approximated by diffusion processes. The reasonableness of this assumption will become clear once we exhibit a

[1] It is assumed that these limits exist.

[2] See [FellerII], page 322.

process and note how its dependence on small increments of time h satisfies conditions (5.1), (5.2), and (5.3).

For the sake of example, we present here the convergence of a sequence of discrete random walks to the most basic of diffusion processes, namely the *Brownian motion.* This example is taken from [Ross], page 184.

We define a random walk on the the real line $\Re$ with cycle time Δt and step size Δx. This means that at each of the discrete points in time $\Delta t, 2\Delta t, \ldots$ the process will move either to the right or to the left by Δx, each possibility occurring with probability $1/2$. The process initially starts off at the origin. Let $X(t)$ be the random variable denoting the position of the process at time t. Then

$$X(t) = \Delta x(X_1 + \cdots X_{\lfloor t/\Delta t \rfloor}),$$

where X_i is determined by the i^{th} step, that is, X_i is either -1 or $+1$, each with probability $1/2$. The X_i are assumed to be independent. Therefore $E[X_i] = 0$ and $E[X_i^2] = 1$ for each i. Thus, observe that

$$\begin{aligned} E[X(t)] &= \Delta x \sum_{i=1}^{\lfloor t/\Delta t \rfloor} E[X_i] \\ &= 0, \end{aligned}$$

and

$$\begin{aligned} \mathrm{Var}(X(t)) &= E\left((X(t))^2\right) \\ &= (\Delta x)^2 \sum_{i=1}^{\lfloor t/\Delta t \rfloor} E[X_i^2] \qquad (5.4) \\ &= (\Delta x)^2 \left\lfloor \frac{t}{\Delta t} \right\rfloor. \end{aligned}$$

Now suppose that one lets both Δx and Δt go to 0. This cannot be done arbitrarily, since the variance (5.4) should go neither to zero, which would imply a deterministic and, in this case, unmoving process, nor to infinity, which would imply complete uncertainty. This says that one must, in the limit, take $\Delta x = c\sqrt{\Delta t}$, for some constant $c > 0$. In that case $E[X(t)] = 0$, while $\mathrm{Var}(X(t))$ converges to $c^2 t$.

The resulting process is a diffusion process known as *Brownian motion.* Observe that the central limit theorem implies that $X(t)$ is normally distributed, with mean 0 and variance $c^2 t$.

A similar limiting procedure may be used to obtain a Brownian motion with non-zero infinitesimal drift.

5.1.2 Expected Convergence Times

In the discrete setting we computed convergence times by setting up a set of linear equations that related the expected convergence times at different states. The coefficients in these linear equations were determined by the transition probabilities. In the continuous setting the analogue of a set of linear equations is a linear differential equation. For diffusions, the coefficients of this linear differential equation are determined by the infinitesimal parameters. Solving the linear differential equation with appropriate boundary conditions yields the expected times to reach some goal. This material may be found in any standard text on diffusions. See for instance [KT2] or [DynYush]. We will focus on time-homogeneous diffusions. This simply means that the transition kernels $Q_{t,\Delta t}$ are independent of t, implying that the infinitesimal parameters are independent of t as well.

Given a diffusion in $\Re^n$ with infinitesimal parameters $\boldsymbol{\mu}(\mathbf{x})$ and $\boldsymbol{\sigma}^2(\mathbf{x})$, one can define a linear operator L, whose coefficients are determined by these parameters. Let us write a point of the state space as $\mathbf{x} = (x_1, \ldots, x_n)$. Correspondingly, we have

$$\boldsymbol{\mu}(\mathbf{x}) = (\mu_1, \ldots, \mu_n),$$

and

$$\boldsymbol{\sigma}^2(\mathbf{x}) = \begin{pmatrix} \sigma_{11}^2(\mathbf{x}) & \ldots & \sigma_{1n}^2(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \sigma_{n1}^2(\mathbf{x}) & \ldots & \sigma_{nn}(\mathbf{x}) \end{pmatrix}.$$

Here $\sigma_{ij}^2(\mathbf{x})$ is the infinitesimal cross-correlation of x_i and x_j, determined by equation (5.2). The second-order linear operator L is then given by

$$L = \frac{1}{2} \sum_{i,j} \sigma_{ij}^2 \frac{\partial^2}{\partial x_i \partial x_j} + \sum_i \mu_i \frac{\partial}{\partial x_i} \tag{5.5}$$

Now consider an open region Ω in $\Re^n$ with boundary $\partial\Omega$. Subject to certain regularity conditions, the expected time to exit the region from a point $\mathbf{x} \in \Omega$ is given by the function $\tau(\mathbf{x})$, where τ satisfies the following partial differential equation and boundary conditions

$$L\tau(\mathbf{x}) = -1, \tag{5.6}$$

$$\text{with } \tau(\mathbf{x}) = 0 \text{ for } \mathbf{x} \in \partial\Omega. \tag{5.7}$$

More complicated boundary conditions may apply for more complicated behaviors. For instance, the boundary may consist of two parts, one of which defines the boundary of the goal ∂G, and the other of which simply specifies the edge of the workspace ∂W. This corresponds in the discrete case to the random walk example of page 124. There we were interested in attaining the origin, while specifying reflection of the random walk at the endpoint a. Similarly, in a continuous domain, one would

specify $\tau(\mathbf{x}) = 0$ for $\mathbf{x} \in \partial G$, and $\frac{d\tau(\mathbf{x})}{d\mathbf{n}(\mathbf{x})} = 0$ for $\mathbf{x} \in \partial W$. Here $\mathbf{n}(\mathbf{x})$ is the outward normal to the boundary ∂W at the point $\mathbf{x}$. Insisting that the normal derivative of τ be zero at the boundary is the manner in which one specifies that the process reflects at the boundary.[3] In general, one cannot specify the boundary conditions arbitrarily. For instance, certain points on the boundary may not be reachable, given the intensity of the expected drift near these points.

Notice that for pure Brownian motion in $\Re^n$, with unit variance, the differential equation (5.6) reduces to a form of Poisson's equation:

$$\nabla^2 \tau = -2, \tag{5.8}$$

with appropriate boundary conditions.

5.1.3 Brownian Motion on an Interval

Solving equation (5.6) can be a formidable task. A common approach is to use the method of Green's functions. However, for some examples the differential equation is easily solvable. One such case is given whenever the coefficients in the operator L are constants. We will look at this case for a diffusion on a subset of the real line, namely the interval $[0, a]$. This example will also demonstrate the relationship of the discrete and continuous cases. Recall the discrete case was analyzed on page 124.

With constant infinitesimal parameters, the one-dimensional diffusion is simply a *Brownian motion with drift*. Let us denote by σ^2 the constant infinitesimal variance, and by μ the constant infinitesimal drift. Note that σ^2 is non-negative, but μ can be any real number. We will assume that the goal is given by the origin, and that reflection occurs at the point a. Thus equation (5.6) becomes:

$$\frac{1}{2}\sigma^2 \tau''(x) + \mu \tau'(x) = -1, \tag{5.9}$$

with boundary conditions $\tau(0) = 0$ and $\tau'(a) = 0$.

First, let us deal with some special cases. If $\sigma^2 = 0$, then the process is deterministic. In particular, the system moves along the real line with velocity μ. If this velocity is strictly negative, then the origin can be attained, otherwise it cannot. Thus, whenever $\mu < 0$, we see that a solution to (5.9) is given by $\tau(x) = -x/\mu$. Notice that in this case the second boundary condition $\tau'(a) = 0$ cannot be satisfied, which is consistent with the fact that (5.9) is now a first-order linear differential equation.

With this special case out of the way, let us assume that $\sigma^2 > 0$. First, let us turn to the case of a pure Brownian motion, with no drift, that is, with $\mu = 0$. In that case, the solution to (5.9) is given by

$$\tau(x) = \frac{1}{\sigma^2} x\,(2a - x).$$

[3] See [DynYush], page 149.

Notice the same quadratic character of the solution that we observed in the discrete setting.

Finally, in the case that $\sigma^2 > 0$ and $\mu \neq 0$, we have that

$$\tau(x) = \frac{x}{-\mu} + \frac{\sigma^2}{2\,\mu^2} e^{\frac{2\mu a}{\sigma^2}} \left(1 - e^{-\frac{2\mu x}{\sigma^2}}\right).$$

Again notice the strong similarity to the discrete case. In particular if μ is negative, then $\tau(x) \leq -x/\mu$. Furthermore, if a is fairly large with $x \ll a$, then $\tau(x) \approx -x/\mu$. In other words, the expected time to reach the origin is essentially the distance to the origin, divided by the expected velocity of approach. Thus, with drift in the correct direction, the diffusion behaves almost like a deterministic process.

We see then that the infinitesimal drift in the continuous setting has strong similarities to the expected velocity at a state in the discrete setting. These similarities carry over to the labelling of states by one-dimensional quantities and to the expected infinitesimal velocity relative to such labellings. In particular, one can transform the state space so that the labelling corresponds to the expected time to attain some goal. In general, given a smooth non-negative labelling that is zero at the goal, if the local drift relative to this labelling is negative at every point and uniformly bounded away from zero, then one can obtain a simple upper bound for the expected time to reach the goal. We will not formally develop these issues in the continuous setting, merely take our lead from the discrete results.[4]

5.1.4 The Bessel Process

A very important diffusion process is the *Bessel process*. This process is a one-dimensional diffusion that measures the distance from the origin of a point undergoing a pure Brownian motion in $\Re^n$. Our interest in this process stems directly from the natural labelling provided by a distance measure. In particular, if one can execute a randomized strategy that makes sufficient expected progress relative to a measure of distance from the goal, then one can be assured of essentially linear convergence times. In other words, the expected convergence times are proportional to the distance from the goal, or better. The simple feedback loop introduced in section 2.4 provides a two-dimensional instantiation of this problem, one which we will examine extensively in the rest of this chapter.

Unfortunately, pure random motions will not make local progress relative to the distance measure in $\Re^n$. We saw this in the discrete setting, and it appears again in the

[4] In order to briefly indicate the relationship between the continuous and discrete settings, suppose that we define the expected infinitesimal velocity relative to some labelling $\ell : \Omega \mapsto \Re$ of the state space as $\nu(\mathbf{x}) = \lim_{h \downarrow 0} \frac{1}{h} E[\Delta_h \ell(\mathbf{X}(t)) \,|\, \mathbf{X}(t) = \mathbf{x}]$. This is the natural analogue of the expected infinitesimal velocity in $\Re^n$. The theory of semi-groups tells us that in fact $\nu(\mathbf{x}) = (L\ell)(\mathbf{x})$, where L is the linear operator (5.5) associated with the diffusion. Thus, if we set $\ell(\mathbf{x}) = \tau(\mathbf{x})$, the expected time to attain the goal, then essentially by definition (equation (5.6)) the expected infinitesimal velocity relative to the labelling must be constant, that is, $\nu(\mathbf{x}) = -1$ for all $\mathbf{x} \in \Omega$. This is precisely the result that we proved in the discrete case. See [FellerII] or [KT2] for a discussion of semi-groups.

continuous setting. The expected convergence times to attain a ball about the origin [5] are on the order of $|\mathbf{x}|^n$. Thus one would need to dilate the state space polynomially in order to see local progress. In order to obtain convergence times that are linear in $|\mathbf{x}|$ one must hope that one's sensors are good enough to overcome the natural outward drift of a randomized strategy. Clearly, this is not always possible. For instance, in the example in section 2.4, for certain approach directions the sensing bias forces the system into a region within which sensing is useless. Only pure randomizing motions are possible. Of course, the strategy is guaranteed eventually to attain the goal, independent of the sensor distribution. One question is whether for sufficiently nice sensors the strategy converges quickly. In answering that question, the Bessel process plays an integral role.

Let us define the Bessel process and exhibit its infinitesimal parameters. We will not derive these parameters nor prove that the Bessel process is indeed a diffusion, but instead refer the interested reader to [KT1] and [KT2]. Later, when we examine the two-dimensional feedback strategy, we will essentially derive the infinitesimal parameters as part of a more complicated derivation.

Let $\mathbf{X}(t) = (X_1(t), \ldots, X_n(t))$ denote a pure Brownian motion in $\Re^n$. Thus the infinitesimal parameters of $\mathbf{X}(t)$ are given by

$$\begin{aligned} \boldsymbol{\mu}(\mathbf{x}) &= \mathbf{0}, \\ \boldsymbol{\sigma}^2(\mathbf{x}) &= \sigma^2 \mathbf{I}_n, \end{aligned}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix.

The Bessel process is given by $\{Y(t), t \geq 0\}$, with

$$Y(t) = \sqrt{\sum_{i=1}^{n} \big(X_i(t)\big)^2}.$$

The infinitesimal parameters of this process are

$$\begin{aligned} \mu_Y(y) &= \frac{n-1}{2y}, \\ \sigma_Y^2(y) &= \sigma^2. \end{aligned}$$

In other words, the infinitesimal variance is the same as for the underlying Brownian motion, but now there is a natural drift away from the origin that is inversely proportional to the distance from the origin.

In deriving these parameters, one approach is to first determine the infinitesimal parameters for the process $Z(t) = Y(t)^2$ from basic principles, then use the following

[5] This assumes that $n > 2$ and that the domain of diffusion Ω is bounded, say, $\Omega = B^n$, the unit ball in $\Re^n$. For $\Re^1$ the expected time is on the order of $|\mathbf{x}|^2$, while for $\Re^2$ it is on the order of $|\mathbf{x}|^2 \log |\mathbf{x}|$, as we have already noted.

fact[6] to obtain the infinitesimal parameters for $Y(t)$. Specifically, if $Z(t)$ is a regular[7] one-dimensional diffusion on some interval in $\Re$ with infinitesimal parameters $\mu_Z(z)$ and $\sigma_Z^2(z)$, and if $g : \Re \mapsto \Re$ is a strictly monotone function with continuous second derivative, then $Y(t) = g(Z(t))$ is a regular diffusion with infinitesimal parameters

$$\mu_Y(y) = \frac{1}{2}\sigma_Z^2(z)\,g''(z) + \mu_Z(z)\,g'(z), \tag{5.10}$$

$$\sigma_Y^2(y) = \sigma_Z^2(z)\left(g'(z)\right)^2, \tag{5.11}$$

where $y = g(z)$.

5.2 Relationship of Non-Deterministic and Probabilistic Errors

Before we are able to analyze the two-dimensional simple feedback loop of section 2.4 for nice error distributions, we must settle on some relationship between the model of non-deterministic error assumed by the strategy and any probabilistic distribution of errors. This will not be as straightforward as one might wish, and we will have to make some arbitrary choices.

Recall that the model of uncertainty assumed by the preimage methodology and by the randomizing example was that of unknown but bounded uncertainty. In other words, actual values were assumed to lie in some uncertainty ball about nominal values, but no particular distribution of errors was assumed. However, in any particular case, the error will be distributed in some specific, although possibly unknown fashion about the nominal value. Consider the case of a probabilistic error distribution. Suppose, for instance, that the non-deterininistic model of error is a sensing error ball of the form $B_\epsilon(\mathbf{x})$. This means that whenever the actual position is $\mathbf{x}$, the sensor returns a value $\mathbf{x}^*$ within distance ϵ of $\mathbf{x}$. Suppose further that the actual error distribution is centered at $\mathbf{x}$. Let p_r be the probability that the observed sensor value $\mathbf{x}^*$ will lie further than distance r from the actual position $\mathbf{x}$ of the system. In symbols, $p_r = \mathbf{P}\{|\mathbf{x}^* - \mathbf{x}| > r\}$. We would like to define the radius ϵ of the non-determinisitc sensing error ball in terms of these probabilities. If there is some r for which $p_r = 0$, then it makes sense to take ϵ to be the infimum of all such r. If $p_r > 0$ for all r, then one may wish to settle on some threshold δ, and take ϵ to be the smallest r for which $p_r < \delta$. Similarly, if $\mathbf{x}^*$ is biased with unknown bias whose magnitude is bounded by $b_{\max}$, then one may first wish to compute r as above assuming no bias, then take ϵ to be $\epsilon = r + b_{\max}$. The same approach applies to other uncertainties, such as control uncertainty.

As an example, consider a two-dimensional sensor with a normal distribution. In particular, suppose the sensor has no bias, that the variances along the two axes

[6] [KT2], page 173.

[7] This means that every point is reachable from every other point. See [KT2].

are identical, and that the measurements along the two axes are uncorrelated. This means that if the actual location is at the origin, then the probability of seeing a sensor value (x_1, x_2) is given by the density function

$$p(x_1, x_2) = \frac{1}{2\pi\sigma^2} e^{-\frac{x_1^2+x_2^2}{2\sigma^2}},$$

where σ^2 is the variance of the measurement along each dimension. A reasonable choice for the radius ϵ of the sensing uncertainty ball might be $\epsilon = 3\sigma$. This corresponds to a certainty threshold of approximately 98.9%.

5.2.1 Control Uncertainty

While this relationship between uncertainty balls and probability distributions seems very straightforward, there are some subtleties. Let us focus first on control uncertainty, then return to sensing uncertainty. Consider again a two-dimensional problem as above, and suppose that whenever one commands a velocity $\mathbf{v}$, the actual velocity[8] is normally distributed, with the error distributions along the two axes being independent and unbiased and having equal variances. This variance will generally be a function of the commanded velocity, so we will denote it as $\sigma^2(\mathbf{v})$. Similarly, within the unknown but bounded model of uncertainty, the actual velocity is assumed to lie within some ball $B_\epsilon(\mathbf{v})$. Here too the radius ϵ is a function of the commanded velocity $\mathbf{v}$. A common approach is to assume that this radius is proportional to the magnitude of the commanded velocity. In short, the unknown but bounded model of uncertainty would say that the actual velocity $\mathbf{v}^*$ satisfies

$$|\mathbf{v}^* - \mathbf{v}| < \epsilon_v |\mathbf{v}|, \tag{5.12}$$

for some constant $\epsilon_v > 0$. Rather than writing $B_{\epsilon_v|\mathbf{v}|}(\mathbf{v})$ for the set of $\mathbf{v}^*$ satisfying constraint (5.12), we will henceforth write velocity uncertainty as $B_{\epsilon_v}(\mathbf{v})$, with the understanding that ϵ_v refers to an error radius that is proportional to the magnitude of the commanded velocity.

In relating the error-ball and probabilistic models as we did above, one might therefore take $3\sigma(\mathbf{v}) = \epsilon$. This says that $3\sigma(\mathbf{v}) = \epsilon_v |\mathbf{v}|$, and hence that $\sigma(\mathbf{v}) = \frac{1}{3}\epsilon_v |\mathbf{v}|$.

We should try to interpret these formal manipulations, and determine whether they make any physical sense. First, let us observe that we have specified uncertainty in terms of velocity, but that we are really interested in position. After all, an action entails executing a velocity for some period of time. Within the error-ball model of uncertainty, an action specifying nominal velocity $\mathbf{v}$ for time Δt means that the change in position is non-deterministically given by $\Delta t\, \mathbf{v}^*$, with $\mathbf{v}^* \in B_{\epsilon_v}(\mathbf{v})$.[9]

[8] This is in free space. In contact space, we modify the velocity as determined by the generalized damper equation (4.1).

[9] For more general error sets, such as non-convex error sets, this is not correct, but generalizations are possible.

Said differently, the change is position is given by Δx, with Δx distributed non-deterministically in the ball $B_{\Delta t \epsilon_v |\mathbf{v}|}(\Delta t\, \mathbf{v})$. Suppose that we now translate this non-deterministic representation into the probabilistic one for an unbiased Gaussian error, in the manner just outlined. In two dimensions this means that the set of position changes is distributed normally about $\Delta t \mathbf{v}$ with standard deviation $\sigma = \frac{1}{3}\epsilon_v |\Delta t\, \mathbf{v}|$.

Looking at this carefully, we should notice a peculiarity in the probabilistic setting. In particular, if instead of commanding velocity $\mathbf{v}$ for time Δt, one repeatedly commands velocity $\mathbf{v}$ for time $\frac{\Delta t}{10000}$, repeating this 10000 times, then one should improve considerably the final accuracy of the desired motion. In particular, the central limit theorem suggests that the final position will be distributed normally about $\Delta t \mathbf{v}$, but now with standard deviation $\sigma/100$. Indeed, if one passes to a diffusion process, that is, to a process in which the motion is commanded repeatedly for infinitesimal amounts of time, the motion actually becomes deterministic. In order to see this, consider the infinitesimal drift and variance. The infinitesimal drift is:

$$\begin{aligned} \boldsymbol{\mu}(\mathbf{x}) &= \lim_{h \downarrow 0} \frac{1}{h} E[\, \Delta_h \mathbf{X}(t) \,|\, \mathbf{X}(t) = \mathbf{x}\,] \\ &= \lim_{h \downarrow 0} \frac{1}{h}\, h\, \mathbf{v} \\ &= \mathbf{v}. \end{aligned}$$

In order to compute the infinitesimal variance, notice that we only need to compute the variance in the x_1 direction (where $\mathbf{x} = (x_1, x_2)$). This is because the variance in the x_2 direction is identical, and because the cross-correlations are zero by independence. Thus, writing $\mathbf{v} = (v_1, v_2)$, we have that

$$\begin{aligned} \sigma_{11}^2 &= \lim_{h \downarrow 0} \frac{1}{h} E[\, \{\Delta_h x_1(t)\}^2 \,|\, \mathbf{X}(t) = \mathbf{x}\,] \\ &= \lim_{h \downarrow 0} \frac{1}{h} \left\{ \frac{1}{9} \epsilon_v^2 h^2 |\mathbf{v}|^2 + h^2 v_1^2 \right\} \\ &= \lim_{h \downarrow 0} h \left\{ \frac{1}{9} \epsilon_v^2 |\mathbf{v}|^2 + v_1^2 \right\} \\ &= 0. \end{aligned}$$

In short, we see that the expected infinitesimal velocity is just the commanded velocity, and that the infinitesimal variance is zero. This implies that the process moves deterministically from $\mathbf{x}$ in the direction $\mathbf{v}$, which does not agree with the non-deterministic error-ball representation. Thus there seems to be a conflict between the two representations. One view is that the problem arises in the non-deterministic model because we have not modelled the velocity error radius as a function of time, but only as a function of the commanded velocity. Another view is that the problem arises in the probabilistic model, at least for Gaussian errors, because the

variance in the change in position is proportional to the square of time. In order to model non-vanishing error, the change in time Δt should appear with at most linear order. A third view is to accept the apparent discrepancy, by realizing that the non-deterministic model may simply conservatively overestimate the motion error. It may indeed be the case that the errors exist as stated both in the non-deterministic and probabilistic cases, but that the non-deterministic model simply does not capture the nice averaging effect that comes into play by the central limit theorem. After all, the non-deterministic model represents a whole collection of possible distributions, including those with biases. For some of these distributions one will see the nice averaging effect, but not necessarily for all.

Nonetheless, this leaves us with a choice as to how we want to represent probabilistic errors once we pass to a diffusion analysis of randomized strategies. One possibility that reconciles the first two explanations above, is to model the error in velocity as white noise. This is a standard approach taken in the study of optimal control (see, for example, [Stengel]). Instead of having an error that grows proportionally to the change in time, one has an error that grows proportionally to the square-root of the change in time. While this implies less error over long motions, it captures the presence of non-zero error over infinitesimal amounts of time, that is, the infinitesimal variance is non-zero. Thus, in terms of our previous representation, the infinitesimal drift is $\mathbf{v}$, which is just the commanded velocity, while the infinitesimal variance is of the form $\sigma_{11}^2 = \sigma_{22}^2 = \sigma^2 > 0$. This says that after $\mathbf{v}$ has been executed for time Δt, the variance in position at that point is on the order of $\Delta t\, \sigma^2$. Relating this to the non-deterministic model, we see that error balls in this case must be modelled as functions of time, with the position error ball at time Δt having radius $\epsilon_v |\mathbf{v}| \sqrt{\Delta t}$.

For simplicity we will stick to the error model that does not capture the time-dependency. This implies that for sufficiently nice velocity distributions, if the commands are issued quickly enough the resulting effect will be a deterministic motion. That may seem to be a bit generous. However, in terms of the diffusion analysis later in this chapter, the significant terms will arise from the variance associated with the guessing of motions rather than from errors in the commanded velocities. Furthermore, once the analysis is complete it will be easy to add in extra terms that capture a non-vanishing infinitesimal variance.

5.2.2 Sensing Uncertainty

A similar problem exists in reconciling the different representations of sensor errors. However, the problem manifests itself slightly differently. In particular, a strategy that employs an unknown but bounded representation of sensing uncertainty may make decisions that differ radically depending on whether the sensed value lies within or beyond some distance of the goal. For instance, in the randomized strategy of section 2.4, if the sensed value lies outside of the position sensing uncertainty of the goal, then the strategy will move towards the goal, while otherwise it will execute a random motion. One immediate problem is that in the probabilistic case the sensor value

may be distributed over an unbounded continuum. However, physical devices usually have a limited range, so the approximation of using a sufficiently large multiple of the standard deviation as the error radius is reasonable. A more pronounced problem is given by the time-dependence of sensor errors. For instance, suppose that a sensor reading is polluted with white noise (or some physically realizable approximation). In a very informal sense, white noise is the time derivative of a Brownian motion. The result of a sensor reading is determined essentially by a random walk in the space of sensor values, but normalized by the time required to obtain the sensor reading. If we imagine that the sensor returns a reading instantly, then the variance in that reading will be infinite. It is only by averaging over a finite extent of time that the sensor value assumes any meaning. However, the variance of the error in this reading is time dependent, and thus so is the radius of an error ball in the non-deterministic representation.

Let us make all of this a little more precise. Let us suppose that a sensor value $\mathbf{s}$ is computed by averaging a white noise process $\{\mathbf{w}(t), t \geq 0\}$ over some small time interval. This means that [10]

$$\mathbf{s} = \frac{1}{\Delta t} \int_{\Delta t} \mathbf{w}(t) dt.$$

Taking expectations, we see that $E[\mathbf{s}] = 0$, while [11]

$$E[\mathbf{s}\mathbf{s}^T] = \frac{1}{\Delta t^2} \int_{\Delta t} \int_{\Delta t} E[\mathbf{w}(t)\mathbf{w}^T(\tau)] dt d\tau.$$

For a Gaussian white noise process, the covariance function is a delta-function, since by definition white noise is completely uncorrelated over time. Thus

$$E[\mathbf{w}(t)\mathbf{w}^T(\tau)] = \mathbf{A}\,\delta(t - \tau),$$

for some constant covariance matrix $\mathbf{A}$. If the noise is symmetric and uncorrelated across different dimensions, then $\mathbf{A}$ is of the form $\mathbf{A} = A\,\mathbf{I}_n$, for some positive constant A. In any event, we therefore see that

$$\begin{aligned} E[\mathbf{s}\mathbf{s}^T] &= \frac{1}{\Delta t^2} \int_{\Delta t} \int_{\Delta t} \mathbf{A}\delta(t - \tau) dt d\tau \\ &= \frac{\mathbf{A}}{\Delta t}. \end{aligned}$$

In short, if the sensing error arises from a white noise process, then the variance in the error depends very much on the timing constants of the sensor. In particular, the longer the averaging process, the better the sensing results. This implies that a strategy that assumes a bounded error ball in making sensor-based decisions must fix

[10] See [Brown], page 254.

[11] If $\mathbf{v}$ is a column vector of dimension n, then $\mathbf{v}^T$ denotes its transpose, and $\mathbf{v}\mathbf{v}^T$ is a matrix of dimension $n \times n$. Thus $E[\mathbf{v}\mathbf{v}^T]$ measures all the covariance terms.

a particular minimum sensing time, in order to be sure that the results fall within that error ball with high enough probability. Said differently, for every sensing error ball, the system tacitly is assuming a certain timing characteristic that makes the error ball valid. One must therefore be careful when making statements that involve small changes in time. These simply are not modelled in the preimage methodology as set forth in chapter 4. Formally one could include time-dependent sensors fairly easily, by modelling both actions and sensory operations as functions of time. This does however complicate the description of preimages since now the response time of the termination predicate plays a role. This path leads into the domain of control theory. We will not follow this path, but simply assume that sensors return values instantaneously.

The previous discussion generalizes to the case of a biased sensor. In this case the maximum magnitude of the unknown bias in the probabilistic model is added to the radius of the bounding error ball of the non-deterministic model. The timing characteristics are not affected by the presence of a bias.

Let us say that the assumption of an instantaneous sensor is reasonable whenever the time interval Δt used to determine the error radius of a sensing uncertainty ball is significantly smaller than any other time interval used in executing a sensor-based strategy. In other words, if all motions are executed for some time interval of considerably greater order than Δt, then one may regard the sensor as instantaneous, ignoring the dependency on Δt. In the upcoming diffusion analysis of a simple feedback loop, this condition is not satisfied, since computing instantaneous expected velocities and variances involves shrinking all time intervals to zero (see equations (5.1) and (5.2)). However, if we take the view that the diffusion analysis approximates a discrete-time process in which the timing constants of the sensors are considerably shorter than all other timing constants, then we may continue to regard the assumption of an instantaneous sensor as reasonable. We will make this assumption, bearing in mind that a more complete analysis should consider a framework in which error balls are time-dependent.

5.3 A Two-Dimensional Simple Feedback Strategy

The tools are now in place for analyzing the strategy outlined in section 2.4 in the special case that the sensing and command errors have unbiased Gaussian distributions. The reason that we would like to analyze the strategy for this special set of sensing errors is to determine how well the strategy behaves when the uncertainty is fairly nicely behaved itself. We know that the strategy will always succeed eventually, independent of the error distributions, so long as these distributions yield the error balls assumed by the strategy. However, one would like the strategy to converge reasonably quickly when the error distributions are nicely behaved. This is because there are well-known optimal control strategies in such cases (see, for instance, [Stengel]). While the randomized strategy suggested in this thesis clearly cannot be optimal, it will nonetheless converge reasonably quickly for a wide range of starting

positions. Thus one can be assured that if the sensors happen to be fairly well behaved, then the strategy will converge quickly, and otherwise it will converge. Thus one does not need to know precisely how the sensors are behaved, but can rely on a general strategy.

The Task

Let us begin by restating the task and the strategy. The task is to attain a disk of radius $r > 0$ centered at the origin of the two-dimensional plane. It is assumed that the goal is recognizable, that is, there is a one-bit sensor that signals goal attainment. Additionally there is a position sensor, which has an error ball with radius ϵ_s. Shortly we will assume that the error distribution is Gaussian, but the statement of the strategy does not assume any particular distribution. The system is assumed to be a first order system, with velocities as commands. The error in the actual velocity executed is likewise assumed to be represented by an error ball of radius $\epsilon = \epsilon_v |\mathbf{v}|$, where $\mathbf{v}$ is the commanded velocity.

The Strategy

The strategy operates as follows. The basic idea is to move towards the origin when doing so will decrease the distance for all possible interpretations of the current sensed position, and otherwise to execute a random motion. We will model the random motion as a Brownian motion, and analyze the whole process as a diffusion. However it should be understood that this is just an approximation to the actual discrete-time process, since the strategy in general will include a delay due both to sensing and motion execution.

It is possible to improve this strategy by taking account of the goal, and of preimages of the goal. In particular, rather than trying to decrease the distance to the origin, a strategy could try to decrease the distance to the goal. Additionally, rather than choosing a completely random motion when it is impossible to decrease the distance to the goal, the strategy could guess between covering backprojections of the goal. We have implemented various simulations of these more knowledgeable strategies, but for our purposes here we will focus on the simple form of the *sensing-guessing* strategy. [The term "sensing-guessing" derives from the strategy's use of both sensor-based motions and random motions, coupled with the view of randomization as a means of guessing the direction to the goal.]

Reducing Distance to the Origin

First, let us determine the conditions under which it is possible to reduce the distance to the goal. Consider figure 5.1. Instead of writing points as $\mathbf{x} = (x_1, x_2)$ we will now write them as $\mathbf{p} = (x, y)$. The sensor value is at the point $(k, 0)$, with $k > 0$. Since only the distance from the origin is of importance, we can assume that the sensor value lies on the x-axis, as in the figure. In this figure it is possible to reduce the distance to the origin for all possible interpretations of the sensor value. This is because the

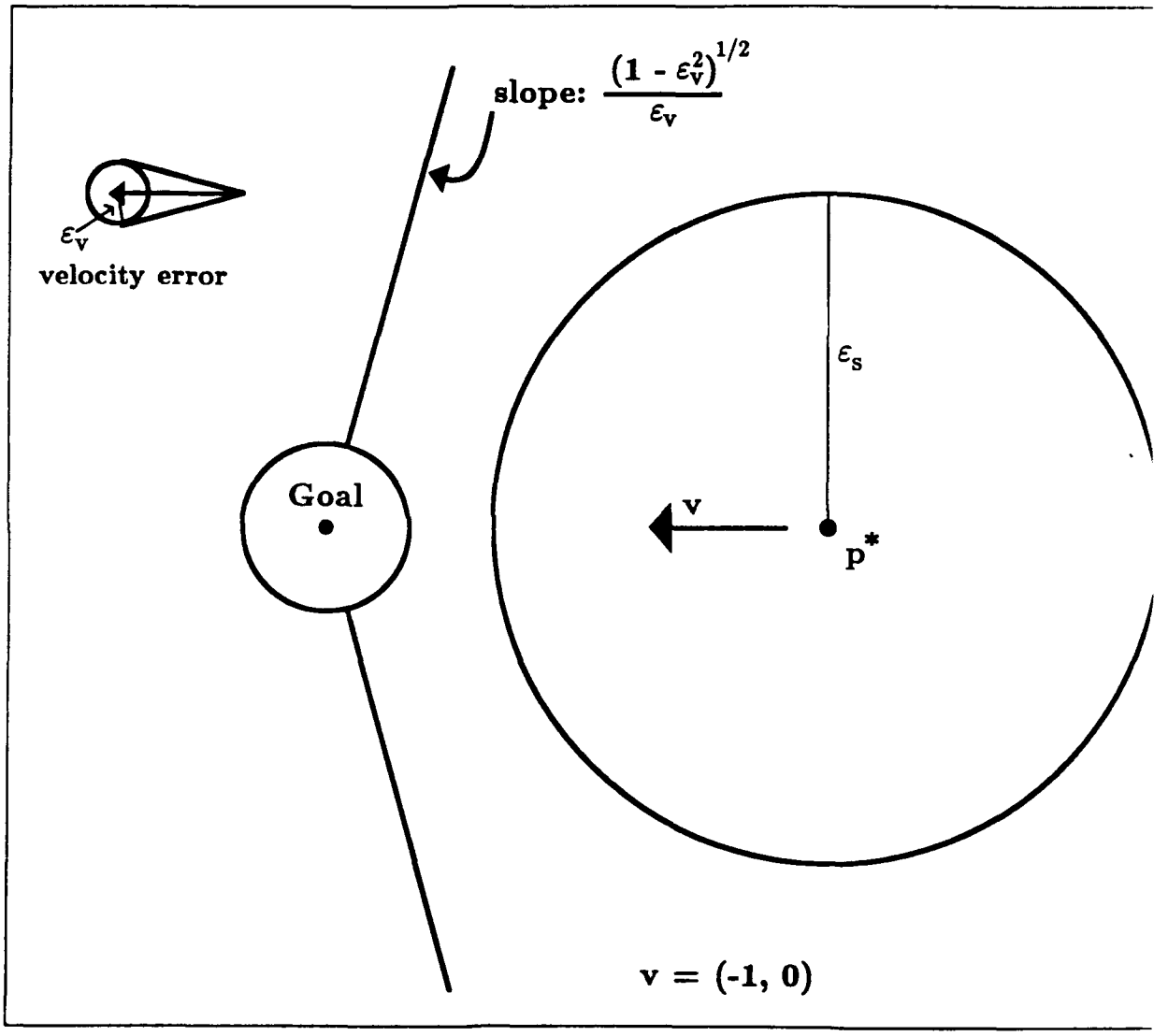


Figure 5.1: For all interpretations of the indicated sensed position, the distance to the goal may be decreased.

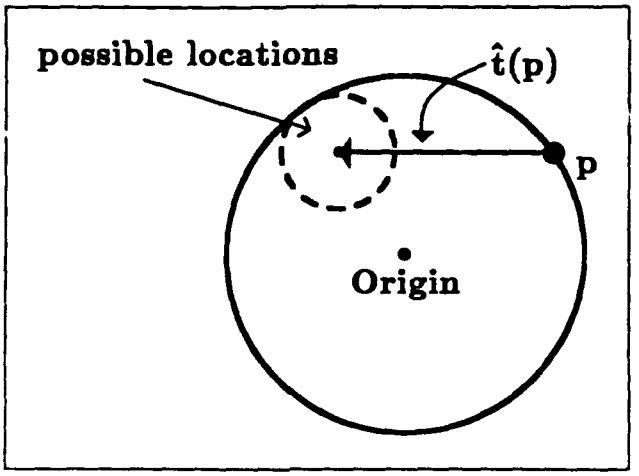


Figure 5.2: This figure shows the maximum time $\hat{t}(\mathbf{p})$ that the velocity $\mathbf{v} = (-1, 0)$ may be executed before the system might move further away from the origin than it is at the start of the motion. The system's start position is $\mathbf{p}$. The disk bounded by the dashed circle represents the possible locations at time $\hat{t}(\mathbf{p})$.

ball of interpretations lies to the right of the two lines passing through the origin with slopes $\pm\sqrt{1-\epsilon_v^2}/\epsilon_v$. [These slopes are determined by the lines bounding the velocity error cone. In particular, $\sin^{-1}(\epsilon_v)$ is just the half-angle of the velocity error cone.] If the sensed position were close enough so that the error ball overlapped the region to the left of these lines, then it would not be possible to reduce the distance to the origin for all possible interpretations of the sensor. In order to see that these lines correctly characterize the condition under which the distance to the origin may be reduced, imagine that the state of the system lies on one of these lines. By symmetry, the commanded velocity will be chosen to be of the form $\mathbf{v} = (-v, 0)$, with $v > 0$. If the velocity uncertainty is given by ϵ_v, then it is possible for the system to move in a direction that is perpendicular to the relevant line. Instantaneously this motion does not change the system's distance from the origin, and thus represents the boundary condition between guaranteed approach towards the origin and possible motion away from the origin.

Maximum Approach Time

Given that a sensed value lies far enough away from the origin that it is possible to reduce the distance to the origin for all possible interpretations, the question arises as to what the commanded approach velocity should be and how long it should be executed. Let us just assume that the commanded velocity has unit magnitude, so that we can focus on the maximum amount of time that the system may execute that velocity without moving further away from the origin. Equivalently, if one fixes

the duration of a motion, then one can adjust the maximum velocity accordingly. Consider now figure 5.2. The figure indicates the effect of an uncertain motion on a particular starting position $\mathbf{p}$. The commanded velocity is $\mathbf{v} = (-1, 0)$. At each instant in time $t > 0$, the set of possible positions is given by the time-indexed forward projection $F_{\mathbf{v},t}(\{\mathbf{p}\})$, which is an open ball of radius $t\,\epsilon_v$, centered at $\mathbf{p} + t\,\mathbf{v}$. So long as this ball lies within the system's starting distance $|\mathbf{p}|$ of the origin, then the motion has reduced the system's distance from the origin. For the sensor interpretation $\mathbf{p}$, the maximum time $\hat{t}(\mathbf{p})$ that the system may execute the motion is thus given by the condition that the forward projection at time $\hat{t}(\mathbf{p})$ just be tangent to the circle of radius $|\mathbf{p}|$ centered at the origin. Minimizing over all possible sensor interpretations of the sensed value $\mathbf{p}^* = (k, 0)$, the maximum time that the system may execute the motion $\mathbf{v} = (-1, 0)$ is thus given by

$$t_{\max}(k) = \min_{\mathbf{p} \in B_{\epsilon_s}(\mathbf{p}^*)} \hat{t}(\mathbf{p}).$$

Now let us determine the maximum time $\hat{t}(\mathbf{p})$ for a given point $\mathbf{p}$. This time satisfies the equation

$$|\mathbf{p} + t\,\mathbf{v}| + t\,\epsilon_v = |\mathbf{p}|. \tag{5.13}$$

Clearly $t = 0$ is a solution to this equation, corresponding to the initial degenerate tangency of the forward projection with the circle of radius $|\mathbf{p}|$. The other solution in t of this equation will correspond to the maximum allowable time that the velocity $\mathbf{v}$ may be commanded, assuming that in the interval in between these two times the inequality $|\mathbf{p} + t\,\mathbf{v}| + t\,\epsilon_v \leq |\mathbf{p}|$ holds. Solving for t by twice squaring equation (5.13), we arrive at four possible solutions. Two of these are zero. The remaining two are given by

$$t = \frac{2}{\mathbf{v} \cdot \mathbf{v} - {\epsilon_v}^2} \left[\pm \epsilon_v \, (\mathbf{p} \cdot \mathbf{p})^{1/2} - \mathbf{p} \cdot \mathbf{v} \right].$$

If $\mathbf{v} = (-1, 0)$, as we have been assuming, and if $\mathbf{p} = (x, y)$, then this becomes

$$t = \frac{2}{1 - \epsilon_v^2} \left[\pm \epsilon_v \sqrt{x^2 + y^2} + x \right].$$

Observe that this really only makes sense if $\epsilon_v < 1$. It is reasonable to thus restrict ϵ_v, since otherwise commanding a velocity $\mathbf{v}$ could in principle cause a motion in any arbitrary direction. Denote the solution corresponding to $\epsilon_v \sqrt{x^2 + y^2} + x$ by t^+, and the solution corresponding to $-\epsilon_v \sqrt{x^2 + y^2} + x$ by t^-. Of these two solutions, one is the solution we are seeking, while the other was merely introduced by our squaring operation. Clearly we want a solution for which $t > 0$, so if we can show that $t^- > 0$, then it is the desired solution since $t^- < t^+$. In order to see that $t^- > 0$, define the function

$$f(t) \;=\; |\mathbf{p}| - |\mathbf{p} + t\,\mathbf{v}| - t\,\epsilon_v$$

$$= \sqrt{x^2+y^2} - \sqrt{(x-t)^2+y^2} - t\,\epsilon_v.$$

Now, $f(0) = 0$, and $f(t^-) = 0$ by definition of t^-. Given that the start position $\mathbf{p}$ lies to the right of the lines of slope $\pm\sqrt{1-\epsilon_v^2}/\epsilon_v$, the possible locations of the system at time t must lie wholly inside the circle of radius $|\mathbf{p}|$, at least for small values of time t. Thus the inequality $f(t) > 0$ holds for small values of t, as desired. This implies that $f'(0) > 0$. Computing the derivative of f, we see that

$$\begin{aligned} f'(t) &= \frac{x-t}{\sqrt{(x-t)^2+y^2}} - \epsilon_v \\ &= \frac{x-t-\epsilon_v\sqrt{(x-t)^2+y^2}}{\sqrt{(x-t)^2+y^2}}. \end{aligned}$$

In particular $\text{sign}(f'(0)) = \text{sign}(x - \epsilon_v\sqrt{x^2+y^2})$. So, we see that $x - \epsilon_v\sqrt{x^2+y^2} > 0$, which says that $t^- > 0$, as we wished to show. [We could also have argued directly that $x - \epsilon_v\sqrt{x^2+y^2} > 0$ since $\mathbf{p}$ lies to the right of the lines of slope $\pm\sqrt{1-\epsilon_v^2}/\epsilon_v$.]

Finally, consider $f'(t^-)$. Since $f(t^-) = 0$, we have that

$$0 \le \sqrt{(x-t^-)^2+y^2} = \sqrt{x^2+y^2} - t^-\,\epsilon_v.$$

This says that

$$\begin{aligned} \text{sign}\left(f'(t^-)\right) &= \text{sign}\left(x - t^- - \epsilon_v\left(\sqrt{x^2+y^2} - t^-\,\epsilon_v\right)\right) \\ &= \text{sign}\left(x - \epsilon_v\sqrt{x^2+y^2} - t^-\,(1-\epsilon_v^2)\right) \\ &= \text{sign}\left(\epsilon_v\sqrt{x^2+y^2} - x\right) \\ &= -1. \end{aligned}$$

In other words, $f'(t^-) < 0$. This says that the solution t^- does indeed describe the maximum duration of the motion.

In short, given that the starting position is $\mathbf{p}$, the nominal velocity $\mathbf{v} = (-1, 0)$ may be commanded throughout the time interval $[0, t^-]$. During that time interval, the system's distance from the origin is guaranteed to be no greater than its starting distance $|\mathbf{p}|$. Furthermore, for any shorter duration than t^-, the system is guaranteed to approach closer to the origin, independent of the actual error distribution of velocities within the error ball about the nominal commanded velocity.

We have computed the maximum time that a motion may be executed for a particular interpretation of the sensor position. Using this we can find the maximum time that is safe for all possible interpretations. Given a sensed position $\mathbf{p}^* = (k, 0)$,

with k positive and far enough from the origin, the maximum amount of time that the velocity $\mathbf{v} = (-1, 0)$ may be commanded is given by

$$\begin{aligned} t_{\max}(k) &= \min_{\mathbf{p} \in B_{\epsilon_s}(\mathbf{p}^*)} \hat{t}(\mathbf{p}) \\ &= \min_{\mathbf{p} \in B_{\epsilon_s}(\mathbf{p}^*)} \frac{-2}{1-\epsilon_v^2} \left(\epsilon_v \sqrt{x^2+y^2} - x \right) \\ &= \frac{-2}{1-\epsilon_v^2} \max_{\mathbf{p} \in B_{\epsilon_s}(\mathbf{p}^*)} \left(\epsilon_v \sqrt{x^2+y^2} - x \right). \end{aligned} \tag{5.14}$$

Here we are writing $\mathbf{p}$ as $\mathbf{p} = (x, y)$.

Let us therefore focus on maximizing the function $q(x, y) = \epsilon_v \sqrt{x^2+y^2} - x$, subject to the constraint that $(x, y) \in B_{\epsilon_s}(\mathbf{p}^*)$. A more sophisticated strategy would only consider those sensory interpretations that lie outside of the goal. This would amount to maximizing $q(x, y)$ subject to the constraint that $(x, y) \in B_{\epsilon_s}(\mathbf{p}^*) - G$, where $G = B_r(\mathbf{0})$ is the goal disk. It is a straightforward matter to modify the strategy accordingly, but we will not do so here.

If $\epsilon_v = 0$, that is, if there is no command error, then $q(x, y) = -x$. Thus $q(x, y)$ is maximized when x is as close to the origin as possible. If the strategy does not take the goal into account in deciding which points need to be moved closer to the origin, but considers the full sensing error ball, then $q(x, y)$ is maximized at $x = k - \epsilon_s$. This is the smallest x-coordinate of a point in the sensing error ball $B_{\epsilon_s}(\mathbf{p}^*)$

Now consider the case $0 < \epsilon_v < 1$. Let us construct the level curves in the plane, given by $q(x, y) = c$, with c some constant. Since $k > 0$ and far enough from the origin, we can assume without loss of generality that $x > 0$. Furthermore, by the same argument that showed that $t^- > 0$ above, we can assume that $c < 0$. Thus we have that

$$\begin{aligned} x + c &= \epsilon_v \sqrt{x^2+y^2}, \\ x^2 + 2xc + c^2 &= \epsilon_v^2 (x^2+y^2). \end{aligned}$$

So

$$(1-\epsilon_v^2)\, x^2 + 2xc + c^2 - \epsilon_v{}^2\, y^2 = 0,$$

from which we see that the level curves are hyperbolas, given by

$$\frac{\left(x + \dfrac{c}{1-\epsilon_v^2} \right)^2}{\left(\dfrac{c\epsilon_v}{1-\epsilon_v^2} \right)^2} - \frac{y^2}{\left(\dfrac{c}{\sqrt{1-\epsilon_v^2}} \right)^2} = 1,$$

with $c < 0$. See figure 5.3. We are interested in the right-hand branch. In particular, we are interested in finding the hyperbola with the maximum c value that touches a point in the sensing error ball about the sensed value $\mathbf{p}^*$. It is clear that the maximum c value is achieved at the boundary of the sensing error ball. Thus we are looking for a hyperbola that is tangent to the circle of radius ϵ_s that is centered at $\mathbf{p}^*$. There are two possibilities.

First, it is possible that the curvature of a hyperbola at its vertex exceeds the curvature of the circle that bounds the sensing error ball. In that case there are two potential tangency points in the first quadrant, that is, there are two locations along the upper right branch of the hyperbola at which a horizontal translation would bring the hyperbola into tangential contact with the circle. One of the potential tangencies occurs at the vertex of the hyperbola on the x-axis. The other tangency occurs somewhere further along the hyperbola. Our aim is to find one such hyperbola that is actually tangent to the circle, and whose associated c value is a maximum. Second, it is possible that the curvature of the hyperbolas is less than that of the sensing error circle. In that case, the only point of potential tangency occurs on the x-axis, and thus the maximizing hyperbola is given by that hyperbola which passes through the point $(k - \epsilon_s, 0)$.

Let us first solve for the tangency condition, then worry about the curvature issue later. Let us assume that the sensing error ball lies strictly inside the wedge determined by the two rays emanating from the origin into the right-half plane with slopes $\pm\sqrt{1-\epsilon_v^2}/\epsilon_v$. This condition is given by $k > \epsilon_s/\sqrt{1-\epsilon_v^2}$. If this condition is not satisfied, then commanding velocity $v = (-1, 0)$ for a non-zero duration of time could potentially increase the distance from the origin for some point in the sensing error ball.

We can write the equations for the circle and the hyperbola as:

$$(x-k)^2 + y^2 = \epsilon_s^2 \qquad \text{and} \qquad \frac{(x-h)^2}{a^2} - \frac{y^2}{b^2} = 1,$$

with $k > 0$ as above, and $h = \dfrac{-c}{1-\epsilon_v^2}$, $a = \dfrac{-c\,\epsilon_v}{1-\epsilon_v^2}$, and $b = \dfrac{-c}{\sqrt{1-\epsilon_v^2}}$. If we eliminate y from these equations we get

$$\epsilon_s^2 - (x-k)^2 = \frac{b^2}{a^2}(x-h)^2 - b^2.$$

In other words,

$$(a^2+b^2)\,x^2 - 2\,(a^2k + b^2h)\,x + [a^2k^2 + b^2h^2 - a^2\epsilon_s^2 - a^2b^2] = 0.$$

So

$$x = \frac{a^2k + b^2h \pm \sqrt{(a^2k+b^2h)^2 - (a^2+b^2)(a^2k^2 + b^2h^2 - a^2\epsilon_s^2 - a^2b^2)}}{a^2+b^2}.$$

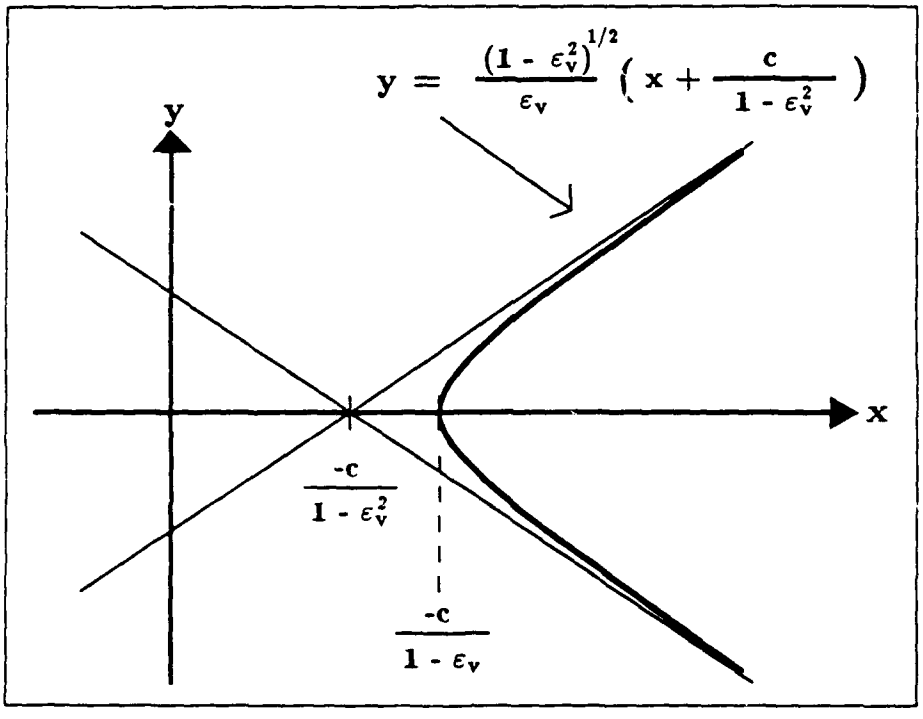


Figure 5.3: The right branch of a hyperbola. The hyperbola is parameterized by c, and represents an iso-value line of the function $q(x,y) = \epsilon_v \sqrt{x^2 + y^2} - x$ described by equation (5.14).

Since we are looking for a tangency point, the discriminant in this solution must actually be zero. This additional constraint allows us to solve for c and thus for the appropriate hyperbola that is tangent to the circle. Thus

$$\begin{aligned} 0 &= a^4k^2 + b^4h^2 + 2a^2b^2hk - a^4k^2 - a^2b^2h^2 + a^4\epsilon_s^2 + a^4b^2 \\ &\quad - a^2b^2k^2 - b^4h^2 + a^2b^2\epsilon_s^2 + a^2b^4 \\ &= 2a^2b^2hk - a^2b^2h^2 + a^4\epsilon_s^2 + a^4b^2 - a^2b^2k^2 + a^2b^2\epsilon_s^2 + a^2b^4 \\ &= -a^2b^2(h-k)^2 + a^2(a^2+b^2)(\epsilon_s^2+b^2). \end{aligned} \tag{5.15}$$

Since $a > 0$, one can divide equation (5.15) by a^2 to obtain

$$0 = -b^2(h-k)^2 + (a^2+b^2)(\epsilon_s^2+b^2). \tag{5.16}$$

If we instantiate the values of a, b, and h, we see that

$$a^2 + b^2 = \frac{c^2}{(1-\epsilon_v^2)^2},$$

and thus equation (5.16) becomes

$$0 = -\frac{c^2}{1-\epsilon_v^2}\left(k + \frac{c}{1-\epsilon_v^2}\right)^2 + \frac{c^2}{(1-\epsilon_v^2)^2}\left(\epsilon_s^2 + \frac{c^2}{1-\epsilon_v^2}\right).$$

Since $c \neq 0$ and $0 < \epsilon_v < 1$, the last constraint may be rewritten as

$$0 = -\left(k + \frac{c}{1-\epsilon_v^2}\right)^2 (1-\epsilon_v^2)^2 + \epsilon_s^2(1-\epsilon_v^2) + c^2,$$

and thus

$$c = \frac{\epsilon_s^2 - k^2(1-\epsilon_v^2)}{2k}. \tag{5.17}$$

This value of c determines the correct hyperbola that is tangent to the boundary of the sensing error ball, assuming that there is a non-trivial tangency. The existence of a non-trivial tangency is determined by the curvature of the hyperbola and the circle. Trivial tangency means that the hyperbola is tangent to the circle at the point $(k-\epsilon_s, 0)$. This implies that

$$c = -(1-\epsilon_v)(k-\epsilon_s). \tag{5.18}$$

Although we will not require it in the sequel, let us determine the condition under which only trivial tangency is possible. See figure 5.4. The circle has curvature $1/\epsilon_s$. Let us compute the curvature of the hyperbola

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1. \tag{5.19}$$

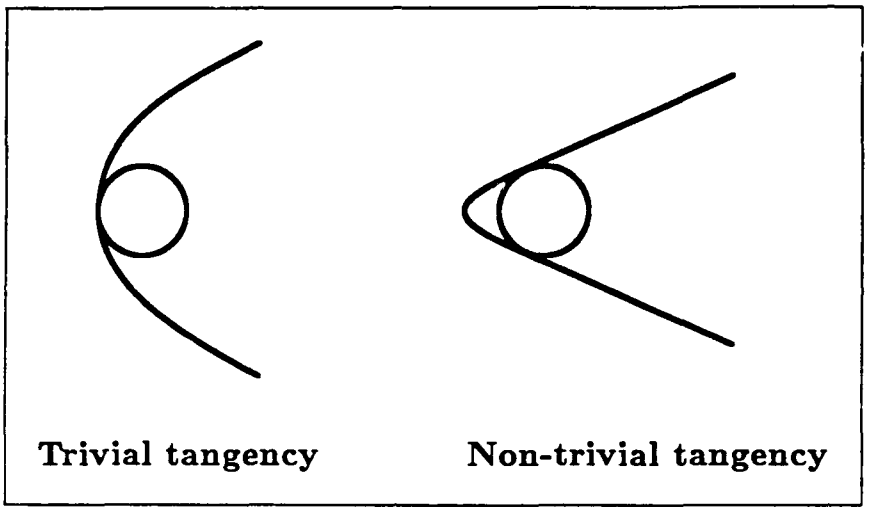


Figure 5.4: The first hyperbola has smaller curvature than the circle, and thus is tangent only in a trivial sense. The second hyperbola has greater curvature, and thus is tangent to the circle in a non-trivial sense.

In general for a curve $y = y(x)$ the curvature κ a a point (x, y) is given by

$$\kappa = \frac{|y''(x)|}{(1 + y'(x)^2)^{3/2}}.$$

For the simple hyperbola (5.19) this expression becomes

$$\kappa = \frac{ba^4}{[(a^2 + b^2)x^2 - a^4]^{3/2}}, \qquad x \geq a.$$

In particular at the vertex $(a, 0)$, the curvature is given by

$$\kappa_{(a,0)} = \frac{a}{b^2},$$

which becomes

$$\kappa = -\frac{\epsilon_v}{c},$$

if we instantiate the values of a and b for the class of hyperbolas that we have been considering. In order for a non-trivial tangency to exist the curvature of the touching hyperbola must exceed the curvature of the circle, that is, $\kappa > 1/\epsilon_s$. If we substitute the maximizing value of c for the touching hyperbola, as given by equation (5.17), we see that this constraint becomes:

$$\begin{aligned} \epsilon_s &> -\frac{c}{\epsilon_v}, \\ \epsilon_s &> \frac{k^2(1-\epsilon_v^2)-\epsilon_s^2}{2\epsilon_v k}, \\ \epsilon_s &> k(1-\epsilon_v). \end{aligned}$$

For the sake of our analysis of the sensing-guessing strategy, let us not worry about whether the maximizing c is determined by equation (5.17) or by equation (5.18). Instead, we will conservatively pick the larger of these. As it turns out, this is always given by (5.17), even when (5.17) does not physically correspond to a hyperbola that has a non-trivial tangency with the sensing circle. In order to see this, consider the inequality that we would like to prove:

$$\frac{\epsilon_s^2 - k^2(1-\epsilon_v^2)}{2k} > -(1-\epsilon_v)(k-\epsilon_s). \tag{5.20}$$

That is:

$$\begin{aligned} \epsilon_s^2 - k^2 + k^2\epsilon_v^2 &> -2k^2 + 2k\epsilon_s + 2k^2\epsilon_v - 2k\epsilon_v\epsilon_s, \\ k^2(1-\epsilon_v)^2 - 2\epsilon_s(1-\epsilon_v)k + \epsilon_s^2 &> 0. \end{aligned}$$

Now consider the function $g(x) = x^2(1-\epsilon_v)^2 - 2\epsilon_s(1-\epsilon_v)x + \epsilon_s^2$. Observe that

$$\begin{aligned} g\left(\frac{\epsilon_s}{1-\epsilon_v}\right) &= 0, \\ g'\left(\frac{\epsilon_s}{1-\epsilon_v}\right) &= 0, \\ g''(x) &> 0. \end{aligned}$$

Thus we see that g is a non-negative function, which establishes the inequality (5.20). We see then that $q(x,y) = c$ is maximized for some c that is bounded from above by the value of c given by (5.17). Thus, in deciding on the maximum amount of time that the velocity $\mathbf{v} = (-1,0)$ may be executed, it is safe to take c to be given by (5.17). This follows from the definition (5.14). We thus have:

$$\begin{aligned} t_{\max}(k) &= -\frac{2}{1-\epsilon_v^2}\,\frac{\epsilon_s^2 - k^2(1-\epsilon_v^2)}{2k} \\ &= k - \frac{1}{k}\,\frac{\epsilon_s^2}{1-\epsilon_v^2}. \end{aligned} \tag{5.21}$$

5.4 Analysis of the Sensing-Guessing Strategy in a Simple Case

We are now in a position to analyze the sensing-guessing strategy outlined earlier (see page 235). We will focus on a particularly nice version of the strategy, in which the sensing and command errors are unbiased and normally distributed. Despite such nice distributions the analysis will quickly become complicated. For this reason most of the results in this section are numerical.

Let us assume that the strategy executes a simple feedback loop that repeatedly senses the current position, then, depending on the distance of the sensed value from the origin, either executes a Brownian motion for a short period of time or reduces the distance to the goal for all possible sensor interpretations. Let us fix the maximum possible time interval between sensing operations as dt. This time interval is used to compute a maximum commanded velocity of approach, analogous to the maximum time computation (5.21). Although the strategy assumes a maximum duration between sensing operations of time dt, we will permit the actual duration to be Δt, with $\Delta t \leq dt$. In a sense, the quantity $1/dt$ serves as a cap on the maximum velocity magnitude that may be executed. This prevents the strategy from becoming a jump process as the time interval Δt shrinks to zero. Instead, the process becomes a diffusion process, and we can use the analysis of this diffusion process to approximately characterize the behavior of the sensing-guessing strategy. [12]

Throughout we assume that sensing is instantaneous, by which we mean that the time constants associated with sensing are much smaller than those of the rest of the system (see section 5.2). While instantaneous sensing could be used to achieve perfect information for the unbiased sensor distribution that we are assuming, the strategy is not actually aware of this distribution. Recall that the strategy should succeed independent of the actual distribution.

The sensing and command errors are assumed to be two-dimensional normal variates with zero bias. We will use a certainty threshold of 98.9% in approximating these errors by uncertainty balls. See again section 5.2. Thus the standard deviation of the sensing error is given by $\sigma_s = \frac{1}{3}\epsilon_s$. Similarly, the standard deviation of the velocity error is given by $\sigma_v = \frac{1}{3}\epsilon_v |\mathbf{v}|$, where $\mathbf{v}$ is the commanded velocity.

Consider now a sensed value $\mathbf{p}^*$ at a distance $k > 0$ from the origin. If $k \leq \epsilon_s/\sqrt{1-\epsilon_v^2}$, then it is not possible to move all interpretations of the sensed value closer to the origin. In this case, the system executes a Brownian motion for time $\Delta t \leq dt$, then takes a new sensor reading. Let us assume that the infinitesimal variance of the Brownian motion is given coordinate-wise by σ_B^2.

If $k > \epsilon_s/\sqrt{1-\epsilon_v^2}$, then the system executes a motion directed towards the origin for time $\Delta t \leq dt$, followed by a new sensor reading. The commanded velocity $\mathbf{v}$ is parallel to the vector $-\mathbf{p}^*$. We can determine the maximum allowable magnitude of

[12] One might very well be interested in a jump process. Indeed, one of the random strategies suggested for the example of section 2.4 was a jump process. However, we will not consider these here.

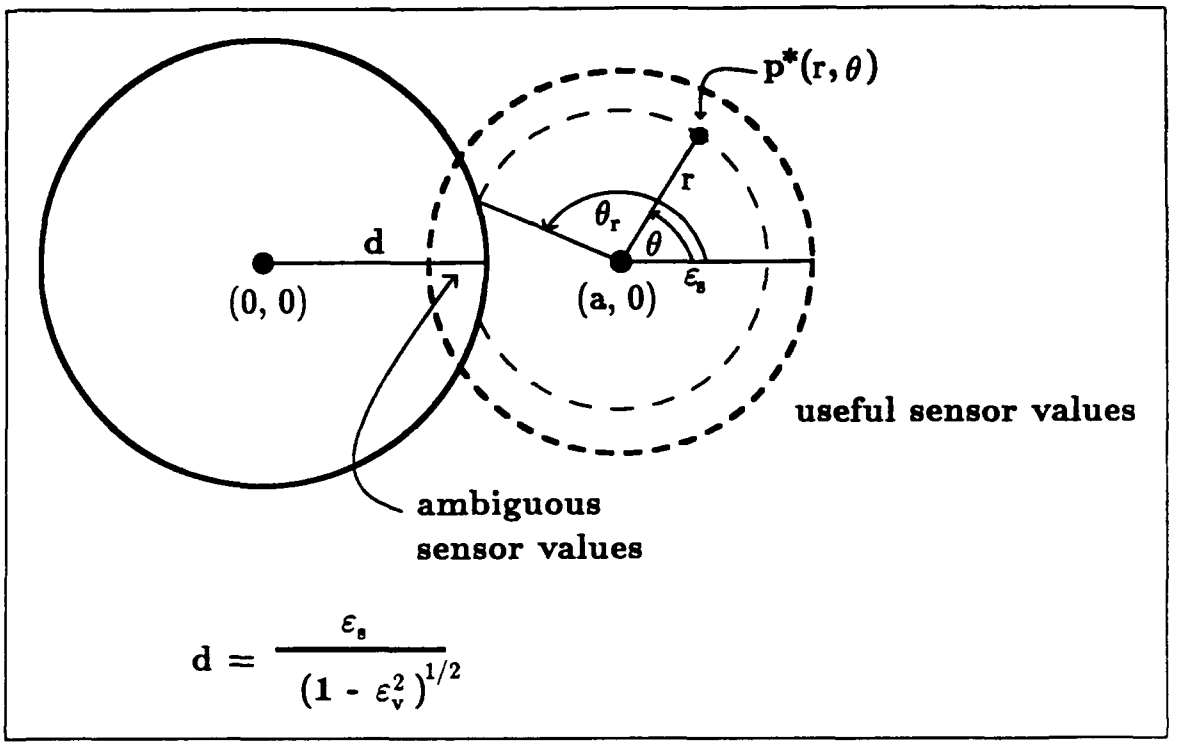


Figure 5.5: The system is at location $(a,0)$. The possible sensor values form a disk of radius ϵ_s about this point. If a sensor value is at least distance d away from the origin, then the system can execute a motion guaranteed to reduce its distance from the origin. The sensor values are shown in a polar coordinate representation (r,θ) relative to the actual position of the system. For each r there is a maximum angle θ_r for which the sensor value lies far enough from the origin. This means that the sensor value $\mathbf{p}^*(r,\theta)$ lies at least distance d from the origin whenever $|\theta| \leq \theta_r$.

this velocity by an argument similar to the one used to establish (5.21). Thus

$$|\mathbf{v}| = \frac{1}{dt}\left(k - \frac{1}{k}\frac{\epsilon_s^2}{1-\epsilon_v^2}\right). \tag{5.22}$$

5.4.1 Expected Progress

Since the problem is radially symmetric, we can assume that the actual position lies on the x-axis at the point $(a,0)$, with $a > 0$. The sensed value $\mathbf{p}^*$ lies (with probability 0.989) in a circle of radius ϵ_s, centered at $(a,0)$.

Expected Change in Position

Let us compute the expected change in position assuming that the sensed value lies far enough away from the origin that the strategy can execute a motion guaranteed to reduce the distance to the origin. Figure 5.5 indicates the portion of the sensing error ball for which the sensed values lie far enough from the origin. By symmetry of the sensing error ball about the x-axis and by symmetry of the velocity error ball, it is clear that the expected change in the y-coordinate of the position $(a, 0)$ is zero. For this reason we will focus simply on the expected change in the x-coordinate. Furthermore, since the velocity error is assumed to be unbiased as well as symmetric, in taking expectations we can simply average over the x-coordinate of all commanded velocities. The averaging here is done with respect to the distribution of the possible commanded velocities, that is, with respect to the distribution of observable sensor values.

In order to compute the expected change in position, let us notice that for each observed sensor value, the system either executes a random motion or a deterministic motion, depending on the distance of the sensed value from the origin. If the observed sensor value lies far enough from the origin, then the expected change in position is simply the commanded velocity times the duration of the motion Δt. We can integrate these commanded velocities over all possible sensor values that lie far enough from the origin, weighting the integrand by the density function that describes the sensor error. The resulting quantity is the expected velocity of the system due to non-randomizing motions. Let us define $\overline{x}(a)$ to be the x-component of this integral. Said differently, $\overline{x}(a)$ is the expected instantaneous change in the x position given that the starting position is at $(a, 0)$ and that the distance of the sensed value from the origin is greater than $\epsilon_s/\sqrt{1-\epsilon_v^2}$, times the probability of actually obtaining a sensor value that far from the origin.

We can write each possible sensor value $\mathbf{p}^*$ in polar form relative to the actual position $(a, 0)$. Specifically, $\mathbf{p}^*(r, \theta) = (x(r, \theta), y(r, \theta))$, where $x(r, \theta) = a + r\cos\theta$ and $y(r, \theta) = r\sin\theta$. We will denote by (r, θ) the velocity command issued when the sensed value is $\mathbf{p}^*(r, \theta)$. Observe also that a sensor value $\mathbf{p}^*(r, \theta)$ is located at a distance $k = k(r, \theta)$ from the origin, where

$$k(r, \theta) = \sqrt{a^2 + r^2 + 2ar\cos\theta}. \tag{5.23}$$

Given the range of possible sensor values $B_{\epsilon_s}(a, 0)$ when the system is at the point $(a, 0)$, and given a disk $B_d(0, 0)$ of radius d centered at the origin, consider the set of sensor values in the set difference $B_{\epsilon_s}(a, 0) - B_d(0, 0)$. These are the set of possible sensor values that are at least distance d away from the origin. If we take d to be $\epsilon_s/\sqrt{1-\epsilon_v^2}$, then this set consists of those sensor values for which the strategy can safely moves towards the origin, that is, for which the strategy can execute a motion guaranteed to reduce the system's distance from the origin, independent of the system's actual location within the sensing error ball $B_{\epsilon_s}(a, 0)$.

Now consider the ring of sensor values at a fixed distance r from the point $(a, 0)$. For some, possibly null, range of angles $(-\theta_r, \theta_r)$, the sensor values $\mathbf{p}^*(r, \theta)$ lie at least

distance d from the origin. See figure 5.5. First let us determine the range of radii r for which this range is non-empty, then let us find an explicit expression for θ_r. Clearly, if the sensing ball about $(a,0)$ lies inside the disk of radius d, then no sensor value lies at least distance d from the origin. Similarly, if the actual location $(a,0)$ lies at least distance d from the origin, then there will be sensor values at all possible radii r that lie at least distance d from the origin. Thus the set $[r_{\min}, r_{\max}]$ of radii for which the interval $(-\theta_r, \theta_r)$ is non-empty is given by:

$$[r_{\min}, r_{\max}] = \begin{cases} \emptyset, & \text{if } a + \epsilon_s \leq d; \\ [0, \epsilon_s], & \text{if } a \geq d; \\ [d-a, \epsilon_s], & \text{otherwise.} \end{cases} \tag{5.24}$$

For a given $r \in [r_{\min}, r_{\max}]$, the angular endpoint θ_r is given by:

$$\theta_r = \begin{cases} \pi, & \text{if } a - r \leq -d \text{ or } a - r \geq d; \\ \cos^{-1}\left(\frac{d^2 - a^2 - r^2}{2ar}\right), & \text{otherwise (assuming } a + r \geq d). \end{cases}$$

The $\cos^{-1}$ function is taken to have values in the range $[0, \pi]$.

The reason for representing the sensor values in terms of polar coordinates relative to the actual location of the system is that the probability density function for the possible sensor values has a simple form in polar coordinates. Specifically, the density function in polar coordinates corresponding to an unbiased two-dimensional normal variate with variance σ^2 is given by:

$$p(r,\theta) = \frac{r}{2\pi\sigma^2} \exp\left\{-\frac{r^2}{2\sigma^2}\right\}, \qquad 0 \leq r < \infty. \tag{5.25}$$

As one would expect, the density function is uniform in θ, that is, it is constant for constant r. Although the function is defined for all non-negative r, we will only consider $r \in [0, \epsilon_s]$, where $\epsilon_s = 3\sigma$. Over this reduced range $p(r,\theta)$ is no longer a density function. However, if the sensor values are indeed constrained to this finite error ball, then one can regain a density function simply by dividing by approximately 0.989 throughout.

The expected instantaneous displacement in the x-direction is thus given by:

$$\overline{x}(a) = \int_{r_{\min}}^{r_{\max}} \int_{-\theta_r}^{\theta_r} v_x(r,\theta)\, p(r,\theta) d\theta dr, \tag{5.26}$$

where $\mathbf{v}(r,\theta) = (v_x, v_y)$ is the commanded velocity determined by $\mathbf{p}^*(r,\theta)$.

Let us expand this formula slightly for the case $dt = 1$. One can simply divide by dt in the general case. Observe that the x-component $v_x(r,\theta)$ of the commanded velocity is of the form:

$$-\left(k - \frac{1}{k}\frac{\epsilon_s^2}{1-\epsilon_v^2}\right)\frac{x(r,\theta)}{|\mathbf{p}^*(r,\theta)|}.$$

Let us focus on the inner integral; call it $\overline{x}_r$. Then we see that

$$
\begin{aligned}
\overline{x}_r &= \int_{-\theta_r}^{\theta_r} - \left(k - \frac{1}{k}\frac{\epsilon_s^2}{1-\epsilon_v^2}\right) \frac{x(r,\theta)}{|\mathbf{p}^*(r,\theta)|}\, d\theta \\
&= 2\int_0^{\theta_r} - \left(k - \frac{1}{k}\frac{\epsilon_s^2}{1-\epsilon_v^2}\right) \frac{x(r,\theta)}{|\mathbf{p}^*(r,\theta)|}\, d\theta \\
&= 2\int_0^{\theta_r} - \left(k - \frac{1}{k}\frac{\epsilon_s^2}{1-\epsilon_v^2}\right) \frac{a + r\cos\theta}{k}\, d\theta \\
&= 2\int_0^{\theta_r} - \left(1 - \frac{1}{k^2}\frac{\epsilon_s^2}{1-\epsilon_v^2}\right) (a + r\cos\theta)\, d\theta \\
&= 2\int_0^{\theta_r} - \left(1 - \frac{1}{(a^2 + r^2 + 2ar\cos\theta)}\frac{\epsilon_s^2}{(1-\epsilon_v^2)}\right) (a + r\cos\theta)\, d\theta \\
&= 2\frac{\epsilon_s^2}{1-\epsilon_v^2}\int_0^{\theta_r} \frac{a + r\cos\theta}{a^2 + r^2 + 2ar\cos\theta}\, d\theta - 2\int_0^{\theta_r} (a + r\cos\theta)\, d\theta \\
&= 2\frac{\epsilon_s^2}{1-\epsilon_v^2}\left[\frac{\theta_r}{2a} + \frac{\mathrm{sign}(a^2 - r^2)}{a}\tan^{-1}\left(\frac{|a-r|}{a+r}\tan\frac{\theta_r}{2}\right)\right] - 2\left[a\,\theta_r + r\sin\theta_r\right].
\end{aligned}
$$

Observe that when a is large, that is, when the system is located far from the origin, the significant term in the expression for $\overline{x}_r$ is $-2\,a\,\theta_r$. Similarly, when a is small, although the terms proportional to $1/a$ now become significant, they tend to be of equal magnitude but opposite sign. Furthermore, for the permissible range of r given by equation (5.24), since $\overline{x}_r$ is negative by construction, these two terms tend to be canceled by the term $-2\,r\sin\theta_r$. Thus again the term proportional to a seems to be the significant term. In short, we see that the sensor essentially acts almost like a spring, pulling the system towards the origin in near proportion to its distance from the origin. This is not completely correct, but it will suffice as a qualitative description.

Finally, the expected drift in the x-direction is determined by integrating over the allowable radii r, that is:

$$\overline{x}(a) = \frac{1}{2\pi\sigma^2}\int_{r_{\min}}^{r_{\max}} \overline{x}_r\, r\, e^{-\frac{r^2}{2\sigma^2}}\, dr.$$

This integral does not admit to a nice explicit description. Instead, we will consider some numerical examples later on. The important observation is that the sensor essentially acts like a spring. As we mentioned in section 5.1.4, pure Brownian motion tends to push a system away from the origin. The question then is whether the pull of the sensor towards the goal is strong enough to overcome the natural push outward due to random motions. Recall that the random motions are required since the system does not know what the error distributions are, but nonetheless should guarantee eventual convergence.

Qualitatively speaking, the inward pull due to sensing is proportional to the distance from the origin, while the outward push due to randomization is inversely proportional to the distance from the origin. Thus there will be a range for which the sensor dominates, and the system moves towards the origin on average. However, as the system approaches close to the origin, eventually the randomization will dominate, and the system will move away from the origin on average. At the boundary between these two modes of behavior, the system moves neither inward nor outward, on average. If the goal is large enough, the system will be sucked into the goal in an almost deterministic fashion. This was the gist of our discussion on local drift. However, if the goal is too small, then the convergence time will become quadratic or worse, as the strategy must rely primarily on random motions rather than on useful sensor readings to attain the goal.

Let us define $p(a)$ to be the probability of obtaining a useful sensor reading whenever the system is at location $(a, 0)$. A useful sensor reading is one for which the system can execute a motion guaranteed to reduce its distance from the origin. Clearly

$$\begin{aligned} p(a) &= \int_{r_{\min}}^{r_{\max}} \int_{-\theta_r}^{\theta_r} p(r,\theta) d\theta dr \\ &= \frac{1}{\pi \sigma^2} \int_{r_{\min}}^{r_{\max}} \theta_r \, r \, e^{-\frac{r^2}{2\sigma^2}} dr. \end{aligned}$$

Suppose that the system is at location $(a, 0)$ and obtains a useful sensor reading $\mathbf{p}^*$. Assume that the system executes a motion determined by equation (5.22) for time Δt. Given this information, the discussion above says that the expected position after execution of the motion, weighted by the probability of actually obtaining a useful sensor reading, is given by:

$$(a + \frac{\overline{x}(a)}{dt} \Delta t, 0).$$

In other words,

$$\begin{aligned} E[\Delta X | \text{useful sensor reading}]\, p(a) &= \frac{\overline{x}(a)\, \Delta t}{dt}, \\ E[\Delta Y | \text{useful sensor reading}]\, p(a) &= 0. \end{aligned}$$

Variance of Positional Change

Let us also compute the variance of the change in each coordinate, assuming that the sensor provides a useful reading. These quantities will enable us to compute the infinitesimal drift and variance in our diffusion approximation to the sensing-guessing strategy.

First, let us suppose that the commanded velocity is $\mathbf{v}$, and let us compute the expectations $E[(\Delta X)^2|\mathbf{v}]$ and $E[(\Delta Y)^2|\mathbf{v}]$. Assume that the velocity is executed for time Δt and that the velocity error is a two-dimensional normal variate with standard deviation $\sigma_v |\mathbf{v}|$. We are assuming as before that $\sigma_v = \frac{1}{3}\epsilon_v$, and that the position error after execution of a motion for time Δt is simply the velocity error scaled by Δt. See the discussion of section 5.2.

Recall that in general, if Z is a random variable, then $E[Z^2] = \mathrm{VAR}[Z] + E[Z]^2$, where $VAR[Z]$ is the variance of Z. This is basically just the definition of the variance of a random variable. In the following expressions, the commanded velocity is of the form $\mathbf{v} = (v_x, v_y)$. We thus have that

$$\begin{aligned} E[(\Delta X)^2|\mathbf{v}] &= \mathrm{VAR}[\Delta X|\mathbf{v}] + E[\Delta X|\mathbf{v}]^2 \\ &= (\Delta t)^2 \sigma_v^2 |\mathbf{v}|^2 + (\Delta t\, v_x)^2 \\ &= (\Delta t)^2 (|\mathbf{v}|^2 \sigma_v^2 + v_x^2). \end{aligned}$$

Similarly,

$$E[(\Delta Y)^2|\mathbf{v}] = (\Delta t)^2 (|\mathbf{v}|^2 \sigma_v^2 + v_y^2).$$

Recall the expression (5.21) for $t_{\max}(k)$. Thinking of k as a function of r and θ as given by equation (5.23), one can write the magnitude of the commanded velocity corresponding to the sensed value $\mathbf{p}^*(r,\theta)$ as $|\mathbf{v}(r,\theta)| = t_{\max}(r,\theta)/dt$. Now let us average over all possible sensor values and associated commands. Then

$$\begin{aligned} E[(\Delta X)^2|\text{useful sensor reading}]\, p(a) &= \int_{r_{\min}}^{r_{\max}} \int_{-\theta_r}^{\theta_r} E[(\Delta X)^2|\mathbf{v}(r,\theta)]\, p(r,\theta) d\theta dr \\ &= \frac{(\Delta t)^2}{(dt)^2} \sigma_v^2 \int_{r_{\min}}^{r_{\max}} \int_{-\theta_r}^{\theta_r} [t_{\max}(r,\theta)]^2\, p(r,\theta) d\theta dr \\ &\quad + \frac{(\Delta t)^2}{(dt)^2} \int_{r_{\min}}^{r_{\max}} \int_{-\theta_r}^{\theta_r} \frac{[t_{\max}(r,\theta)]^2\, [x(r,\theta)]^2}{|\mathbf{p}^*|^2}\, p(r,\theta) d\theta dr. \end{aligned}$$

We can, for appropriate definitions of $I(a)$ and $I_x(a)$, write this as

$$E[(\Delta X)^2|\text{useful sensor reading}]\, p(a) = \frac{(\Delta t)^2}{(dt)^2} \sigma_v^2 I(a) + \frac{(\Delta t)^2}{(dt)^2} I_x(a). \tag{5.27}$$

Similarly,

$$E[(\Delta Y)^2|\text{useful sensor reading}]\, p(a) = \frac{(\Delta t)^2}{(dt)^2} \sigma_v^2 I(a) + \frac{(\Delta t)^2}{(dt)^2} I_y(a). \tag{5.28}$$

Here $I_x(a) + I_y(a) = I(a)$.

The important observation is that these expectations are proportional to $(\Delta t)^2$. This means that if we pass to a diffusion approximation, the infinitesimal variances will be zero. This is because one divides by Δt in computing the infinitesimal parameters, then allows Δt to approach zero. The fact that the infinitesimal variances approach zero means that the portion of the sensing-guessing strategy that results from useful sensor values is essentially a deterministic process. This is due to our assumption that the velocity error scales with Δt, rather than with $\sqrt{\Delta t}$ (see the discussion in section 5.2). If instead we assumed that the velocity error was due to white noise, then it would scale with $\sqrt{\Delta t}$. In that case the expressions (5.27) and (5.28) above would be slightly different. Specifically, the coefficient of $I(a)$ would now be proportional to Δt rather than $(\Delta t)^2$. In passing to a diffusion approximation, this says that the infinitesimal variance contains a term proportional to $I(a)$. It is straightforward to perform the inner integral with respect to θ in the definition of $I(a)$. Again, the outer integral with respect to r has no explicit representation. We will not perform the integration here, but simply mention that the integral contains a term proportional to a^2, as one would expect.

Infinitesimal Parameters of an Approximating Diffusion Process

Having determined the expectation and variance of the change in position given that the system obtains a useful sensor reading, let us now compute these quantities in the general case, that is, for arbitrary sensor readings, assuming that the system is at location $(a, 0)$ and has just taken a sensor reading. Recall that the variance of the Brownian motion is σ_B^2. Recall further that $p(a)$ is the probability of obtaining a useful sensor reading when the system is at the location $(a, 0)$.

$$E[\Delta X] = E[\Delta X|\text{useful sensor reading}]\,p(a) + E[\Delta X|\text{Brownian motion}]\,(1 - p(a)). \tag{5.29}$$

$$\text{So}\quad E[\Delta X] = \frac{\overline{x}(a)}{dt}\,\Delta t. \tag{5.30}$$

$$E[\Delta Y] = 0. \tag{5.31}$$

Similarly,

$$E[(\Delta X)^2] = \frac{(\Delta t)^2}{(dt)^2}\left[\sigma_v^2\, I(a) + I_x(a)\right] + (1 - p(a))\left[\Delta t\,\sigma_B^2 + o_x(\Delta t)\right],$$

$$E[(\Delta Y)^2] = \frac{(\Delta t)^2}{(dt)^2}\left[\sigma_v^2\, I(a) + I_y(a)\right] + (1 - p(a))\left[\Delta t\,\sigma_B^2 + o_y(\Delta t)\right],$$

where $o_x(\Delta t)$ and $o_y(\Delta t)$ contain terms of order less than Δt. It follows that the infinitesimal drift and variance of an approximating diffusion process derived from the sensing-guessing strategy are given by:

$$\begin{aligned}\boldsymbol{\mu}(a,0) &= \left(\frac{\overline{x}(a)}{dt},\, 0\right),\\ \boldsymbol{\sigma}^2(a,0) &= \begin{bmatrix}\sigma_G^2(a) & 0\\ 0 & \sigma_G^2(a)\end{bmatrix},\end{aligned}$$

where

$$\sigma_G^2(a) = (1 - p(a))\,\sigma_B^2.$$

We should also verify that the higher-order infinitesimal moments vanish, but this follows in a straightforward manner from the results for Brownian motions.

A Radial Process

The behavior of the sensing-guessing strategy is radially symmetric, since we have assumed that the sensing and control errors are symmetric. We can thus think of the strategy as a one-dimensional process on the positive real line. We will approximate the actual sensing-guessing strategy by a diffusion process. Specifically, define $D(t) = \sqrt{X^2(t) + Y^2(t)}$, where $(X(t), Y(t))$ is the position of the system at time t. Then $D(t)$ is the distance from the origin at time t. In determining the infinitesimal parameters of $D(t)$ we will use an argument very similar to the one used to establish the infinitesimal parameters of the Bessel process (see section 5.1.4 and [KT2]).

Define, first of all, $Z(t) = X(t)^2 + Y(t)^2$. So $D(t) = \sqrt{Z(t)}$. As usual, we shall write $X(t + \Delta t) = x + \Delta X$, where $x = X(t)$ is given at time t, and Δt is the time between sensing operations. Thus ΔX is a random variable. A similar notation is used for Y and Z. ΔX and ΔY are independent random variables. Modulo terms of order less than Δt, both of these random variables have essentially normal distributions, with variance $\Delta t \sigma_G^2$. Given that $(x, y) = (a, 0)$, $E[\Delta X]$ and $E[\Delta Y]$ are given by (5.30) and (5.31) above. Observe that

$$\begin{aligned}\Delta Z &= X^2(t + \Delta t) + Y^2(t + \Delta t) - x^2 - y^2\\ &= x^2 + 2x\,\Delta X + (\Delta X)^2 + y^2 + 2y\,\Delta Y + (\Delta Y)^2 - x^2 - y^2\\ &= \left[(\Delta X)^2 + (\Delta Y)^2\right] + 2\left[x\,\Delta X + y\,\Delta Y\right].\end{aligned}$$

By symmetry, we can assume without loss of generality that $(x, y) = (a, 0)$. Thus

$$\begin{aligned}E[\Delta Z] &= E[(\Delta X)^2] + E[(\Delta Y)^2] + 2\,x\,E[\Delta X] + 2\,y\,E[\Delta Y]\\ &= \Delta t\,\sigma_G^2 + \overline{\sigma}_x(\Delta t) + \Delta t\,\sigma_G^2 + \overline{\sigma}_y(\Delta t) + 2a\,\frac{\overline{x}(a)}{dt}\,\Delta t,\end{aligned}$$

where $\overline{o}_x(\Delta t)$ and $\overline{o}_y(\Delta t)$ contain terms of order less than Δt.

This tells us that the infinitesimal drift for the process Z is

$$\mu_Z(a^2) = 2a\frac{\overline{x}(a)}{dt} + 2\sigma_G^2.$$

Let us now compute the terms for the infinitesimal variance of ΔZ. The computation makes use of the fact that the higher order infinitesimal moments for the processes X and Y vanish, and the fact that ΔX and ΔY are independent.

$$\begin{aligned}
E[(\Delta Z)^2] &= E[(2x\,\Delta X + (\Delta X)^2 + 2y\,\Delta Y + (\Delta Y)^2)^2] \\
&= E[4x^2(\Delta X)^2 + (\Delta X)^4 + 4y^2(\Delta Y)^2 + (\Delta Y)^4 + 4x(\Delta X)^3 + 4y(\Delta Y)^3 \\
&\quad + 8xy\Delta X\Delta Y + 4x\Delta X(\Delta Y)^2 + 4y\Delta Y(\Delta X)^2 + 2(\Delta X)^2(\Delta Y)^2] \\
&= 4\,E[x^2(\Delta X)^2 + y^2(\Delta Y)^2] + o(\Delta t) \\
&= 4\sigma_G^2\, z\,\Delta t + o(\Delta t),
\end{aligned}$$

where $o(\Delta t)$ contains terms of order less than Δt, that is, terms proportional to $(\Delta t)^p$, with $p > 1$. We see then that the infinitesimal variance of Z is given by

$$\sigma_Z^2(a^2) = 4\sigma_G^2 a^2.$$

Furthermore, one can argue that the higher-order infinitesimal moments vanish, since they vanish for the underlying Brownian motion processes.

In order to determine the infinitesimal parameters of the process D, we will use equations (5.10) and (5.11) from page 229, with $g(z) = \sqrt{z}$. Thus

$$\begin{aligned}
\mu_D(a) &= \frac{1}{2}\sigma_Z^2\left(-\frac{1}{4}\frac{1}{a^3}\right) + \mu_z\left(\frac{1}{2}\frac{1}{a}\right) \qquad (5.32) \\
&= \frac{1}{2}4\sigma_G^2 a^2\left(-\frac{1}{4}\frac{1}{a^3}\right) + \frac{2a\,\overline{x}(a)\,(1/dt) + 2\sigma_G^2}{2a} \\
&= \frac{\overline{x}(a)}{dt} + \sigma_G^2\frac{1}{2a} \\
&= \frac{\overline{x}(a)}{dt} + \frac{(1-p(a))\,\sigma_B^2}{2a}. \qquad (5.33)
\end{aligned}$$

This expression says that the infinitesimal drift consists of two terms, one pulling the system towards the origin, the other pushing it away. The inward pull is due to the sensor, while the outward drift is due to randomization. This outward pull arises in the same manner as it did for the Bessel process.

Next, observe that

$$\begin{aligned}\sigma_D^2(a) &= \sigma_Z^2 \left(\frac{1}{2}\frac{1}{a}\right)^2 \\ &= \left(4\,\sigma_G^2\,a^2\right)\frac{1}{4a^2} \\ &= \sigma_G^2 \\ &= (1-p(a))\,\sigma_B^2.\end{aligned}$$

In other words, the infinitesimal variance is the same for the radial process as it is coordinate-wise for the two-dimensional representation of the sensing-guessing strategy.

Finally, let us observe that if the error arising from motions commanded in response to useful sensor values is non-vanishing in the limit as Δt goes to zero, then one must add another term to the expression for σ_Z^2. This term is proportional to the integral $I(a)$. The term carries over to the expressions for μ_D and σ_D^2, effectively adding another outward pull to the radial drift. This outward pull is essentially proportional to the distance from the origin. Intuitively it arises from command errors in much the same way that an outward drift arises from purely random motions.

The important observation to take from (5.33) is that the two terms have opposite sign. Thus, there is some point a_0 for which $\mu_D(a_0) = 0$. If $a > a_0$, then the net radial drift is negative, meaning that on average the system is moving towards the origin. Conversely, if $a < a_0$, then the drift is positive, meaning that on average the system is moving away from the origin. This says that if the goal radius r is bigger than a_0, then the system behaves almost like a deterministic process, moving towards the goal with expected approach velocity greater than or equal to $\mu_D(r)$. Thus the expected convergence time is essentially bounded by $-a_s/\mu_D(r)$, where a_s is the starting location of the system. On the other hand, if $r < a_0$, then the system will act very similar to a Brownian motion process, randomly walking about inside the annulus $r \leq a \leq a_0$ until the goal is attained. The convergence times now become slightly worse than quadratic in the distance from the origin.

5.4.2 An Example

Solving for a_0 is in general a difficult task, since the expression (5.33) involves several integrals that have no explicit analytic description. We will therefore consider a simple numerical example. Suppose that the error parameters are given as follows.

Sensing Error:	ϵ_s	$= 7$
Velocity Error:	ϵ_v	$= 0.5$
Brownian Motion Variance:	σ_B^2	$= 1.0$

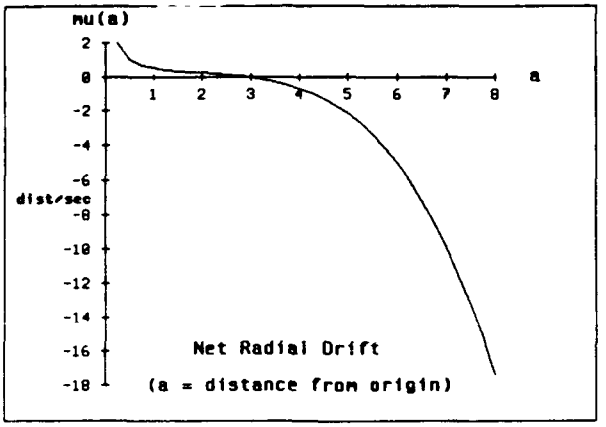


Figure 5.6: Effective radial drift for the sensing-guessing strategy.

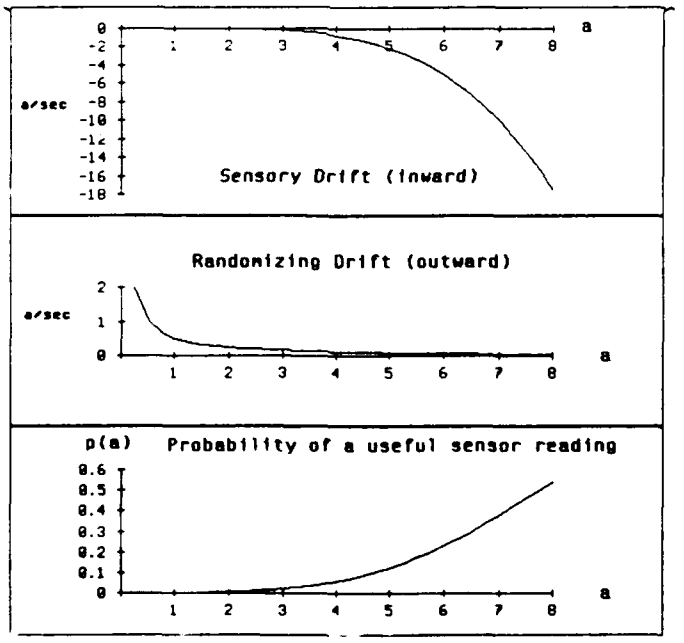


Figure 5.7: Component drifts for the sensing-guessing strategy, along with the probability of obtaining a useful sensor reading.

Figures 5.6 and 5.7 indicate the resulting radial drift. Figure 5.7 resolves this drift into the inward pull due to the sensor and the outward push due to randomization. The figure also shows how the probability of obtaining a useful sensor reading increases with the distance from the goal. The figures indicate that the value a_0 at which the drift switches from negative to positive is around $a = 3.0$. It is interesting to note that the value of a_0 is considerably less than ϵ_s for this example. In order for a sensed value to be useful it has to lie outside the circle of radius $d = \epsilon_s/\sqrt{1-\epsilon_v^2}$. In this example $d \approx 8.1$. In order to guarantee that a sensed value will lie far enough from the origin, one would have to insist that the system be at least distance $d + \epsilon_s \approx 15.1$ from the origin. Thus a strategy that wished to guarantee entry into the goal in a fixed number of motions could do so only if the radius of the goal was at least 15.1. However, a randomized strategy can guarantee eventual entry. Indeed, for the nice Gaussian sensor distribution that we have assumed, sufficiently many sensor values lie outside the circle of radius d, that the expected approach velocity points towards the origin whenever the system is at least distance $a \approx 3$ from the origin. The difference between $d + \epsilon_s \approx 15.1$ and $a \approx 3$ shows quite dramatically how a randomized strategy can extend the convergence region of a goal beyond that provided by a bounded-step guaranteed strategy.

We should add our usual caveat to these observations. The strategy could be considerably improved for the particular pair of sensing and control errors assumed in the analysis above. For instance, by always assuming that the sensor value $\mathbf{p}^*$ is correct, and issuing a commanded velocity of the form $\mathbf{v} = -\mathbf{p}^*/dt$, the expected approach velocity could be made to point towards the goal for all positions of a, not just for $a > 3$. This is because the sensing error has no bias. However, as we have stated before, the strategy was designed to succeed for all error distributions consistent with the bounds ϵ_s and ϵ_v, not just unbiased Gaussian errors. A strategy that always interpreted the current sensed value as correct could easily converge to the wrong location. This difficulty was demonstrated in figure 2.7 for a sensor with a fixed but unknown bias. Thus we have employed a strategy that is suboptimal in the presence of unbiased Gaussian errors, but that still converges reasonably quickly, and more importantly, that converges for all possible error distributions.

Convergence Times

Let us examine the expected convergence times for the current example. In section 5.1.2 we discussed a differential equation that models the expected convergence time of a diffusion process. We can solve this equation numerically to obtain estimates for the convergence times of the sensing-guessing strategy for various goal radii.

Figure 5.8 displays the numerical solution to the differential equation (5.6), assuming that the goal is located at $a = 5$, and that the system reflects at $a = 12$. The expected times to reach the goal seem to satisfy a downward-opening quadratic. This is not surprising, given the spring-like behavior of the sensor. After all, the expected approach velocity at a given point is almost proportional to the distance from the origin. For these examples, the maximum cap on velocity magnitude was

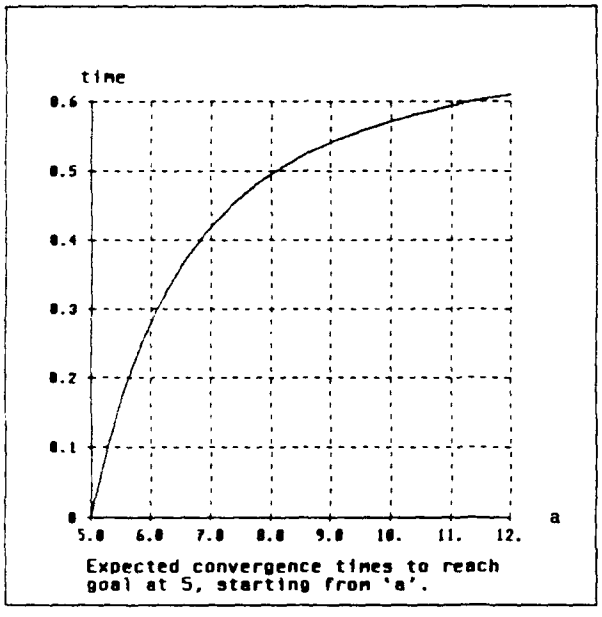


Figure 5.8: Expected times to reach a goal of radius 5 from different starting locations, for the sensing-guessing strategy.

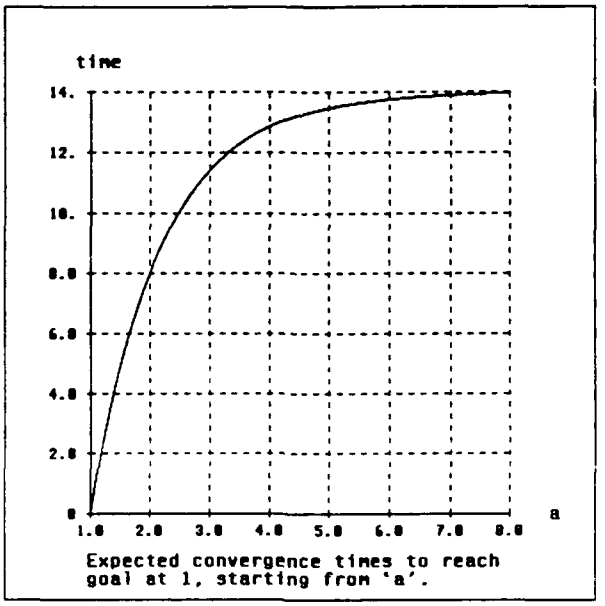


Figure 5.9: Expected times to reach a goal of radius 1 from different starting locations, for the sensing-guessing strategy.

indirectly given by using $dt = 0.1$. So, at a distance of a units from the origin, the expected approach velocity is no greater than a/dt units per second, that is, $10\,a$. In fact, it is often much less because not all sensor values provide a useful sensor reading. For instance at $a = 8$, the expected approach velocity is approximately -17.3 (see figure 5.6), whereas at $a = 12$ it is about -66.1.

The quadratic nature of the convergence times may seem to contradict the claim that the convergence times are linear in the distance from the origin. In fact there is no such contradiction, since the linearity claim is simply an upper bound on the convergence times. Since the expected approach velocity does increase with the distance from the origin, one would expect the actual convergence times to be considerably less than the predictions made by the linearity bound. Indeed, if one erected a line tangent to the curve of figure 5.8 at the point $a = 5.0$, this line would represent the linear upper bound. The downward-opening nature of the curve reflects the fact that the actual performance is considerably better.

A visually more convincing argument is made by considering the convergence times for a goal radius r that lies inside the radius a_0. Recall that a_0 is the location at which the expected approach velocity switches sign. In some sense a_0 represents an attraction point, since locally the expected infinitesimal velocity points towards a_0. Thus if a goal has a smaller radius than a_0, then convergence is guaranteed by the variance of the Brownian motion, not by the motions suggested by the sensor.

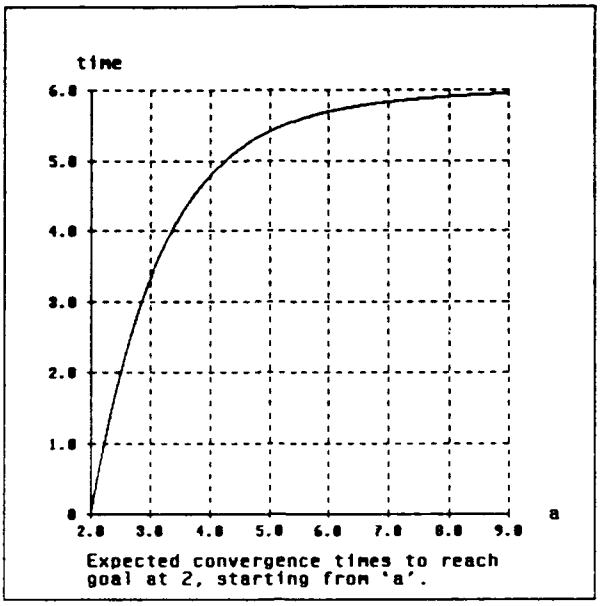


Figure 5.10: Expected times to reach a goal of radius 2 from different starting locations, for the sensing-guessing strategy.

The greater the variance of the Brownian motion, the faster the convergence. For the case, $r = 1.0$, figure 5.9 shows the expected convergence times, assuming reflection at $a = 8$. The curve is again similar to a quadratic, but the convergence times are one to two orders of magnitude greater than they were for the case $r = 5$. Notice that the segment from $a = 8$ to $a = 5$ appears nearly linear with respect to the scale of the entire curve from $a = 8$ to $a = 1$. In other words, relative to the scale of this problem, where the goal is at $r = 1$, the convergence times of the previous problem, where the goal was at $r = 5$, are indeed nearly linear.

Finally, figure 5.10 displays the convergence times for another problem in which $r < a_0$. In this case $r = 2$. Again, the times are considerably greater than for the case $r = 5 > a_0$. Furthermore, comparing this figure to figure 5.9, one sees how dramatic is the difference between moving from $a = 3$ to $a = 2$ and moving from $a = 2$ to $a = 1$.

5.4.3 Simulations

We tested the sensing-guessing strategy in simulation. The results agree qualitatively with those obtained from the analysis above. In particular, for the case in which the goal radius is 5, and the starting location is at the point $(12, 0)$, the average time to attain the goal, averaged over 1000 trials, was approximately 0.505. The maximum

and minimum times to attain the goal were 0.039 and 2.64, respectively, and the experimentally obtained standard deviation was 0.365. The numerical results from the data for figure 5.8 suggested an expected convergence time of approximately 0.61 in this case.

Similarly, for the case $r = 1$, with a starting location at $(8,0)$, the average time to attain the goal was 9.14, with a standard deviation of 7.86. The minimum and maximum times were 0.116 and 58.2. These statistics were also obtained from 1000 trials. The numerical results from the data for figure 5.10 suggest an expected convergence time of approximately 14 in this case.

The simulation statistics and the analytical/numerical predictions do not agree, except in terms of order of magnitude. Part of this is due to the fact that we assumed a pure diffusion process for the analytical results, whereas the simulations were implemented as discrete-time processes, with a time step that was on the order of dt. As a consequence, the variance arising from command errors became significant. Recall that we assumed that the variance in the command error disappears as the time step approaches zero. A larger variance implies that the system is more likely to make big motions, which can decrease convergence times. Nonetheless, as a first approximation to the qualitative behavior, the numerical results describe the sensing-guessing strategy reasonably well. Indeed, upon taking $\Delta t = dt/100$, there was a marked improvement in the results. For the case $r = 5$, the average over 1000 trials was 0.582. For the case $r = 1$, the average over 1000 trials was over 11.

Biases

If we add biases to the sensing or control errors, then the problem is no longer symmetric. In particular, the infinitesimal drift and variance depend not only on the distance from the origin but on the exact location $\mathbf{p} = (x, y)$. The differential equation (5.6) describing the expected time to attain the goal is thus a two-dimensional partial differential equation. Rather than solve this equation explicitly or numerically, let us try to obtain a qualitative description of the behavior of the system.

We will focus on sensing biases. That is because a sensing bias can radically change the convergence properties of a region near the goal. In particular, as we shall see, a point in state space may change from a point at which sensing is good enough to move the system towards the goal on the average, into a point at which only randomization is possible. While velocity biases can also change convergence properties, the feedback strategy of this chapter was designed to make progress for all velocities in the velocity error cone. Thus the change affected by a velocity bias manifests itself primarily as a small change in the direction (and magnitude) of the infinitesimal drift. Locally this does not change the convergence properties of points near the goal, assuming that the velocity bias is small. The velocity bias clearly may have a global effect since changing the local infinitesimal drift changes the natural paths of the system. The analysis of such global changes goes beyond the scope of this thesis.

It may be useful to consider again the example of section 2.4. In that example the

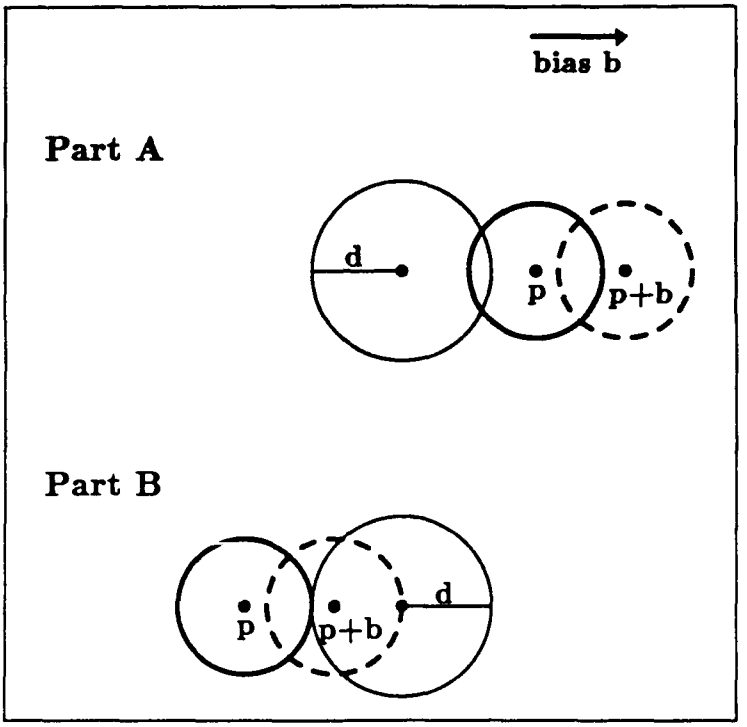


Figure 5.11: This figure indicates the effect of a sensing bias on the usefulness of sensor values. A sensor value is useful if it lies outside the circle of radius d. In each of Part A and Part B the actual state of the system is at the point $\mathbf{p}$. The solid circle about $\mathbf{p}$ indicates the range of possible sensor values without any bias. The dashed circle about the point $\mathbf{p} + \mathbf{b}$ indicates the actual range of sensor values, assuming a bias $\mathbf{b}$. In Part A the bias increases the range of useful sensor values, while in Part B the bias decreases the range of useful sensor values.

sensing error was given by a constant bias. The effect of this bias was to facilitate goal attainment from certain approach directions, while preventing it from others. If one introduces sensing biases into the simple feedback strategy of the current chapter, the effect is similar. Effectively the bias shifts the sensing uncertainty ball. For some states of the system this means that the observed sensor values are shifted away from the origin, thus increasing the likelihood that the system will obtain a useful sensor value. For other states, the observed sensor values are shifted towards the origin, thereby preventing the system from knowing in which direction to move.

First, imagine that the system is unaware of a bias in the sensing uncertainty. Instead, the simple feedback loop operates as before on the assumption that the sensing error ball has radius ϵ_s and that the velocity uncertainty is given by ϵ_v. Let $d = \epsilon_s/\sqrt{1-\epsilon_v^2}$. Recall that this means that whenever the system observes a sensor value that lies at least distance d from the origin, then it will execute a

motion guaranteed to make progress towards the origin. If the sensed value lies within distance d of the origin, then the system executes a randomizing motion. Now, let $\mathbf{p}$ be the actual state of the system and let $\mathbf{b}$ be the unknown bias in the sensor. See figure 5.11. If there were no bias, the range of possible sensor readings would be given by a ball of radius ϵ_s centered at $\mathbf{p}$, that is, by $B_{\epsilon_s}(\mathbf{p})$. With the bias $\mathbf{b}$, the range of possible sensor values is shifted by the bias, that is, it is given by the ball $B_{\epsilon_s}(\mathbf{p}+\mathbf{b})$.

In short, the behavior of the feedback loop at the point $\mathbf{p}$, assuming an unknown bias $\mathbf{b}$, is the same as it would have been at the point $\mathbf{p}+\mathbf{b}$ for a feedback loop in which there is no sensing bias. In particular, the local infinitesimal drift at the point $\mathbf{p}$ in the biased case is the same as it would be at the point $\mathbf{p}+\mathbf{b}$ in the unbiased case. Suppose that $\mathbf{p}$ and $\mathbf{b}$ are actually parallel, as in figure 5.11. Then in the biased system the expected velocity of approach is increased at the point $\mathbf{p}$ whenever $\mathbf{p}\cdot\mathbf{b} > 0$, and is decreased otherwise. Thus a system approaching the origin from a direction on the opposite side of the origin relative to the bias must quickly resort to randomization. If the bias is reasonably small relative to the size of the goal then this is not a permanent problem. Eventually, as the system drifts around the goal, the sensing bias begins to facilitate goal approach, and the system is again able to rely on sensing to make progress towards the goal. (See again the example of section 2.4 that deals with the case of sensing error due purely to a fixed but unknown bias.)

Thus far we have assumed that the system is unaware of the existence of a bias. If in fact the maximum possible magnitude $b_{\max}$ of the bias $\mathbf{b}$ is known to the system, but not the actual direction, then a safe strategy is to augment the effective sensing uncertainty radius from ϵ_s to $\epsilon_s + b_{\max}$. This increases the value of the safe distance d. As a result, the range of useful sensor values at any state is reduced. This means that the infinitesimal drift towards the origin is decreased in magnitude. Indeed, for some states, sensing may no longer be of any use.

In summary, we see that a sensing bias changes the convergence properties of points near the goal. In particular, there are preferred directions of approach, namely those that are roughly on the same side of the goal as the direction given by the sensing bias. If the sensing bias is small, then the system can safely ignore the bias. If the bias is large, then its maximum magnitude should be incorporated into the decision loop that ensures safe progress towards the goal.

5.5 Summary

This chapter analyzed in detail a simple feedback loop. The task consisted of moving a point in the plane into a circular region at the origin, in the presence of control and sensing uncertainty. Such a task might correspond to the problem of inserting a peg into a hole by sliding the peg on a surface surrounding the hole. The strategy was stated without assuming any particular form of error distribution. Both the control and sensing uncertainty were merely represented as bounded error balls.

The strategy consisted of a combination of sensor-based motions and random motions. Repeatedly, the system would sense its current position, then execute a

motion for a short duration of time. Whenever the sensed position was sufficiently far from the origin, the system would execute a motion guaranteed to reduce its distance from the origin for all possible interpretations of the sensed position. Otherwise, the system would execute a random motion. The purpose of the random motion was to move either to a location from which the sensor could again provide useful information or to attain the goal fortuitously.

The randomized strategy was formulated to succeed independent of the actual error distributions, so long as these distributions satisfied certain bounds. The randomizing aspect of the strategy ensures this success. The convergence time of the strategy, however, depends intimately on the actual error distributions. The strategy was analyzed for a particularly nice pair of error distributions, namely unbiased Gaussian errors in both sensing and control. This analysis involved modelling the behavior of the strategy as a diffusion process. The resulting diffusion approximation determined a range of goal radii for which the strategy converged quickly. In contrast, a strategy that must guarantee entry into the goal within a fixed number of steps would consider the problem unsolvable for many of these goals, namely those goals with small radii. One may conclude that randomization offers a reasonable approach for extending the class of solvable tasks beyond those considered solvable by bounded-step guaranteed strategies.

Chapter 6

Conclusions and Open Questions

6.1 Synopsis and Issues

Randomization and Task Solvability

The main goal of this thesis has been to demonstrate how randomized strategies can extend the class of tasks considered to be solvable. The basic idea is to place a loop around a set of strategies, each of which is guaranteed to accomplish a task if certain preconditions are satisfied. The purpose of the loop is to repeatedly choose and execute one such strategy, in the hope of eventually choosing a strategy that will actually attain the goal. In making its choice, the system executes a strategy whose preconditions are satisfied, should the system ever be fortunate enough to knowingly satisfy the preconditions of some strategy. Generally, however, the preconditions may be too stringent to be satisfied knowingly. In that case, the system randomly selects a strategy. Eventually the system will guess correctly and accomplish its task.

Synthesizing Randomized Strategies

The thesis developed a formalism for generating guaranteed plans to include randomizing decisions and actions. Of particular interest were tasks for which there existed strategies that would locally make progress on the average, relative to some progress measure defined on the system's state space. It was shown that any strategy whose behavior may be modelled as a Markov chain inherently defines a progress measure relative to which it makes progress. The complementary problem, of finding a useful progress measure for a given task, is more difficult. Sometimes distance provides a natural progress measure, but generally a strategy will only make progress on some subset of the state space for such a progress measure. An interesting question is whether it is possible to transform a task description into a progress measure from which one can build a fast randomized strategy. In general one suspects that this problem is no easier than the problem of finding guaranteed strategies or optimal strategies. However, for certain classes of tasks an advantage may be gained by viewing the task in terms of progress measures and nominal plans. This is an open

area.

More Extensive Knowledge States

Although the thesis developed the randomizing formalism in some generality, the specific examples considered were essentially assembly operations involving the attainment of two-dimensional regions, assuming position sensors and first-order dynamics. An interesting project would be to include force information in the randomizing decisions, to make extended use of history, and to consider more complicated tasks. A related question is whether anything is to be gained by defining progress measures on the space of knowledge states. This is the natural setting for such measures once a strategy retains history in making decisions.

Reducing Brittleness

One view of randomized strategies is that they provide a means for reducing the sensitivity of a task solution to initial conditions. After all, the whole approach is based on not knowing exactly which preconditions are satisfied. This view may be carried further to include other parameters of the system. The example of section 2.4 showed how the sensitivity to sensing biases could be avoided by executing randomizing motions, albeit at the cost of increased convergence time. Other parameters, such as the shape and location of objects in the environment, or the specification of the dynamics, may also be subject to uncertainty. It is desirable to construct strategies that need not know precisely the values of these parameters. Donald's [Don89] work on model error forms a natural domain in which to explore randomizing approaches for dealing with uncertainties in the task specification itself. See also [KR]. An interesting open question is whether it is possible to build general strategies from simple and incomplete task descriptions. Randomization may provide part of the answer via its ability to blur the sensitivity to detail.

6.2 Applications

Chapter 2 discussed some of the intended applications of randomized strategies. The assembly and manipulation of objects, mobile robot navigation, and the design of parts and sensors are broad domains of applicability. Let us now relate some of the results of the thesis to these domains.

6.2.1 Assembly

A Formal Framework For Existing Applications

Randomization plays an important role in assembly operations. Randomization appears naturally in the form of noise, both in sensing and control. Furthermore, it is sometimes added purposefully in the execution of assembly strategies. Vibrating

a part in order to overcome stiction is a common example. Spiral searches to locate some feature, while implemented deterministically, are similar to randomization both in their intent, namely to compensate for unknown initial conditions, as well as in their execution, due to control uncertainty. Finally, vibratory bowl feeders actively make use of randomization by purposefully tossing an improperly oriented part back into the bottom of the bowl. The intent is to obtain probabilistically a better orientation of the part on its next pass through the bowl's orienting track.

We see therefore that randomization is a useful tool present in the solution of established manipulation and assembly tasks. One contribution of this thesis has been to provide a formal basis for the use of randomization. In particular, the thesis developed a framework for synthesizing randomized strategies. Within this framework randomization may be viewed as simply another operator, along with the operators of sensing and action. All three operators are essential to the solution of general assembly tasks.

Utilizing Available Sensors

One of the themes of the thesis was to explore the conditions under which progress towards task completion is possible on average. We implemented a peg-in-hole task using a simple camera system to sense position, and we analyzed the convergence properties of a simple feedback loop with a position sensor subject to unbiased Gaussian error. In both cases the task strategy would make use of the position sensor when the sensor provided information that permitted progress towards the goal, and otherwise the strategy would execute a random motion. This combination of sensing and randomization allowed the task to be solved probabilistically under conditions for which no bounded-step guaranteed strategy existed. Not only did the randomization ensure eventual convergence, but for a wide range of initial conditions the sensing information ensured that the convergence was actually rapid.

The moral to be taken from these examples is that a position sensor can provide considerably more information than is made use of in a bounded-step guaranteed strategy. While this information cannot always be interpreted correctly in a guaranteed sense, the combination of randomization and sensing can in many instances naturally sort out the useful from the useless sensor readings. For instance, by randomizing its position, a system can compensate for unknown sensing biases, and in some instances naturally position itself actually to take advantage of the biases.

Using Additional Sensors

Ultimately one should explore more complex sensors. In particular, it is clear that force sensors are useful in disambiguating contact conditions. [Sim79] points out that the extra information to be gained from position sensors by using probabilistic techniques, such as Kalman filters, produces estimates with the same order of magnitude in precision as the sensors themselves. In contrast, two orders of magnitude of improved precision are usually required to meet standard clearance ratios of

assembled parts. By adding force sensors one can enhance greatly the net sensing precision. In terms of randomized strategies and simple feedback loops, this barrier to the improvement in the precision of position sensors alone makes itself visible in the direction of the expected motion of the system. Ultimately, as the system begins to operate below the resolution of its sensors, the randomizing aspect of the strategy dominates the sensing information, and the system drifts away from the goal on average. An unexplored question is how the addition of force sensors could be used to improve the convergence of a randomizing feedback loop.

Eventual Convergence in the Context of Grasping

Despite the advantage of better sensors in terms of improved precision, the sensors can sometimes be difficult to interpret. For instance, consider a multi-fingered hand equipped with torque sensors at each of several tendon-controlling motors. A set of torque readings from these sensors may be difficult to map back onto an interpretation in the world. Fortunately, better sensors are not required in a strict sense to ensure goal attainment. Randomization ensures eventual goal attainment. [This assumes that the randomization is so chosen as to cover the space of interest in finite time, and that the goal is recognizable]. Again, the point is that a randomized strategy makes use of sensing information when it can, but does not stop cold once this information ceases to be useful. This is an important property.

In the multi-fingered hand example, the task might consist of grasping a part stably. If the positions of the fingers relative to the part are not known precisely, or if the dynamic properties of the part itself are not known precisely, then it may not be possible to grasp the part stably on a first attempt. For instance, the center of mass might be in an unexpected location. While one can imagine a series of test operations based on force information to ascertain the dynamic properties of the part, such a battery of tests may not be feasible, due perhaps to a lack of sensors or an inability to interpret them. If this is the case, a simpler strategy might consist of grasping the part by randomly selecting a grasp configuration from a set of grasp configurations, where the set has been chosen to contain the desired but unknown grasp. Although the robot may drop the part a few times, eventually it will select the correct grasp configuration, and the task will be accomplished.

From a practical point of view this discussion suggests that one need not rely heavily on complicated sensors. We know from the work on sensorless manipulation (see [Mas85]) that task mechanics and predictive ability can often be used to solve tasks well below the resolution of available sensors. The thesis suggests that another approach is to use randomization.

Some Assembly Tasks

Some other classes of tasks in which randomization is useful include:

- **Parts Orienting.** Many parts, in particular, polyhedral parts, will assume one of a small set of configurations when dropped onto a tabletop under the

influence of gravity. One approach for orienting a part is to drop it onto a table, then perhaps shake the table or the part until the part winds up in the desired configuration. The advantage of this approach is that it reduces the sensing and manipulation requirements of the orienting system. Instead of being required to orient the part from a possibly arbitrary configuration, the system need simply be able to randomize the part's configuration sufficiently to ensure that the desired orientation is achieved. Additionally, the system need simply be able to recognize the part in its goal orientation. The disadvantage of this approach is that it may require a long time to succeed if the desired orientation is one that occurs infrequently when the part is dropped. More work is required on investigating the usefulness of this approach. Again, we mention the vibratory bowl feeder as a paradigm similar to this approach for orienting parts. [See [BRPM] in this context.]

In terms of the thesis these operations correspond to the nearly sensorless tasks discussed in section 3.13. Sensing is used mainly to signal goal attainment, while randomization is used to ensure eventual convergence. It is up to a planning system that understands the mechanics of the domain, in this case the dynamics of dropped parts, to suggest a sufficient set of randomizing motions.

- **Fine Motions.** One of the applications of randomization is in the final phase of a complex operation. Generally the available control and sensing system will be good enough to complete the gross motion operations of the task, but the fine motions may be difficult to control or observe. A simple example in the human domain is given by the task of opening an electric car window to a desired width in order to adjust the airflow to the rear passengers to a comfortable level. It is impossible generally to position the window precisely on a single attempt. Indeed, the precise opening may not even be known ahead of time. By randomly moving the window back and forth about the desired opening, one can quickly open the window properly.

  Another example is given by the adjustment of interior wall sections during the construction of a house. Once a wall segment has been erected vertically, it is nearly impossible to execute any precise motions. This is because the wall is wedged tightly between the ceiling and the floor. Nonetheless, precise motions are required to ensure that the wall segment is oriented properly in the vertical and horizontal directions. The standard approach is to tap portions of the wall with a large hammer, then consult a scale or plumb to determine the orientation of the wall segment. The effect of the tapping operations is to produce a random walk about the desired orientation. The scale or plumb plays the role of a sensor that serves both to indicate the desired direction of motion as well as to signal goal attainment.

  Within the domain of assembly of nearly-rigid parts there are numerous examples that share common characteristics with theses two examples from the

human world. Tapping parts that are slightly wedged is a common operation. Another common operation is searching for a pin or hole prior to a mating task.

The results of this thesis suggest that goal convergence is rapid if progress towards the desired set point can be made on average. Goal convergence in this case means attaining some small region about the set point. If we take the simple feedback loop of chapter 5 as a guide, one approach for obtaining average progress is to execute motions whose magnitude is nearly proportional to the sensed distance from the goal. This corresponds to the intuitive idea of moving quickly towards the goal when one is far away, and moving slowly otherwise. In the window example one modulates the time interval during which the window is being either opened or closed, while in the wall-tapping example one modulates the impulse of the tapping operations. Once the window is near the desired opening or the wall is nearly vertical, then it may become difficult to control the velocity of the system finely enough to ensure average progress. As in the feedback example of chapter 5, once the system is close to the goal, it effectively relies entirely on randomization to attain the goal.

6.2.2 Mobile Robots

An important characteristic of mobile robots is their existence in an uncertain world. Not only is the robot's initial model of the world incomplete or inaccurate, but the world itself is changing as people and objects move about. Uncertainty is thus a fundamental characteristic of the mobile robot domain.

There is considerable room for work in applying randomizing techniques to mobile robots. Promising areas include navigation, map building, and feature recognition.

Randomization for navigation can help reduce the knowledge requirements of a robot. Robots that use local algorithms in making decisions about global navigation may become trapped in some deterministic state or cycle of states. Randomization can prevent this trap from persisting forever. Even locally this may be useful, for instance when a robot finds itself in a tight corner, unable to determine the proper direction to turn in order to escape. Another example, taken from probabilistic broadcast networks, is given by the problem of several identical robots meeting at the intersection of two or more hallways. If right of way rules are unclear or inapplicable it makes sense to arbitrate these right of way rules by randomization. Each robot simply executes a strategy that randomly and repeatedly tries to proceed through the intersection or gives way for another robot to proceed.

Randomization may be of use in map building, by weakening the requirement for accurate maps. This is a difficult area of research, with potentially promising results. A possible approach is to view a map as one would a noisy sensor reading. Some portions of the map provide clearly useful information, while others do not. Randomization is used to compensate for the incomplete or inaccurate portions of the map. This is an application of randomization as a means of blurring environmental details. As a trite example, suppose that a robot is unsure which offices along a

hallway house graduate students and which house professors. Indeed, the state of the offices might actually be in flux over time. A map might nonetheless contain enough information to depict the topology of the office building as well as the ratio of graduate students to professors. The robot could then use this information to randomly select an office in such a way as to maximize the probability of encountering a professor. A similar problem is given by the task of finding a free Xerox machine in a building in which there are varying numbers of machines on each floor, not all of which are necessarily free or working. This is a classic problem out of decision analysis.

The examples listed so far are fairly simple and at a high level. However they have their counterparts within the internal implementation of the robot. Indeed, one problem with robot systems is the fusion of multiple sensory information. This is often a complicated process, particularly if one of the sensors is at the limit of its range of applicability. For instance, a sonar sensor may indicate the presence of an obstacle in front of the robot, which an infrared sensor may not see. One possibility is simply to arbitrate between the sensors in a random fashion. In short, the robot imagines the presence or absence of certain features in the environment, based on randomly chosen sensory information. There are issues involved here in deciding how often to arbitrate, and whether it is even safe to arbitrate randomly. These are precisely the issues addressed by the planning methodology presented in this thesis. In particular, the connectivity assumption of section 3.2.7 addresses the safety issue. The backchaining process using the operator SELECT of section 3.9 addresses the issue of when to randomize. However, much work remains in mapping these general techniques into the mobile robot domain.

6.2.3 Design

The design of parts and sensors stands as a task complementary to the task of planning assembly motions. Clearly the design problem is much less constrained; *a priori* the space of possible designs has an enormously large number of degrees of freedom. However, much can be learned by considering how particular assembly strategies succeed or fail. The class of randomized strategies provides another clue to the efficient design and usage of parts and sensors.

Sensor Design: A Sensor Placement Example

As an example consider again a random walk on a two-dimensional grid. As we learned in chapter 3 the natural tendency of the random walk is to move away from the origin whenever it is positioned on one of the axes of the grid. More generally, a continuous random walk in a higher dimensional space has a natural tendency to drift away from a goal region situated at the origin. Taking the two-dimensional random walk as an example, suppose that we installed a couple of one-bit sensors on the axes of the grid. These might be implemented as light beams parallel to the grid axes. Then one could reduce the two-dimensional random walk to a pair of one-dimensional random walks. Recall that in a one-dimensional random walk the

average motion progress of the system is zero, rather than away from the goal. Thus if there is any additional sensing, the system will naturally move towards the goal on average.

Specifically, one would let the system perform a two-dimensional random walk until it crossed one of the light beams. Since the light beams cover two lines, the system effectively behaves as if it were performing a one-dimensional random walk with a goal recognizer at the origin. Upon observing that a light beam has been crossed, and remembering which one, the system can then perform a one-dimensional random walk along the appropriate axis until the goal at the center of the two-dimensional grid is attained. The reliability with which the system can perform the one-dimensional random walk depends of course on the control uncertainty. All the sensing in the world is of no use if the control uncertainty is bad enough. However, assuming reliable control but possibly poor sensing, this example demonstrates how an understanding of the capabilities of a randomized strategy may be used for designing sensor placements.

Generalizations of this example involve the reduction of higher dimensional random walks to a series of one-dimensional random walks either by the addition of sensors in appropriate locations or the modification of strategies.

Parts Design

We have already alluded to the design of part shapes in the discussion of part orienting by dropping. An understanding of the dynamic properties and stable resting configurations of differently shaped objects is essential to the design of parts shaped for assembly. Randomization provides a context in which to consider these dynamic properties. Said differently, randomization provides a means of assessing the natural motions of a part. This information is useful for it describes the possible motions of a part in the presence of control error.

Extending the analysis of natural part motions in order to actually design parts is still an open area. The study of randomization as a means of facilitating this process should be a fertile area of future research.

A design criterion related to the notion of natural behavior is made evident by the implementation of the peg-in-hole task and by the example of section 2.4. In these cases, randomization helped the system find a path or region from which progress towards the goal was rapid. This success was possible in these examples because of the system's ability to approach the goal from an arbitrary direction. Generally, that might not be possible. However, by considering the manner in which a system uses information, the regions in which it randomizes its motions, and the regions of fast convergence, a designer can determine whether or not a system will naturally gravitate towards regions of fast convergence. This analysis can then be used to redesign the system if necessary.

6.3 Further Future Work

We have indicated above numerous areas in which randomization may prove fruitful. Let us now briefly indicate some very specific topics in the thesis that deserve further attention.

6.3.1 Task Solvability

One of the motivations for this thesis was to work towards an understanding of the class of tasks solvable by different repertoires of actions. The appeal of randomized strategies lies in their simplicity and pervasive presence. We have shown that randomized strategies can increase the class of solvable tasks over those solvable by bounded-step guaranteed strategies. We have also indicated the manner in which randomization can facilitate task solutions, even when bounded-step guaranteed strategies exist. Nonetheless, there is still missing a language in which one can talk about task solvability and compare different repertoires of actions. Even more difficult is the actual characterization of tasks and strategies in terms of each other. Much work remains to be done in this area.

6.3.2 Simple Feedback Loops

Conditions of Rapid Convergence

We analyzed a randomized simple feedback loop for the two-dimensional task of attaining a circular region in the plane. The strategy was formulated in general terms. However, the results that we obtained indicating fast convergence were numerical results that assumed particular uncertainty values. While the qualitative behavior of the system is similar for varying uncertainty values, it is desirable to obtain a set of explicit conditions formulated in terms of arbitrary uncertainty variables that characterize the regions of fast convergence. Part of the difficulty in determining these conditions is that one of the integrals defining the expected progress of the feedback loop does not possess an analytic closed-form solution. It may be useful in elucidating these conditions to consider lower or upper bounds for this integral.

Biases

The analysis of the simple feedback loop assumed unbiased Gaussian errors. This simplified the problem to a one-dimensional problem formulated in terms of the distance of the system from the origin. We discussed the qualitative behavior of the system once biases are introduced. Again, it would be useful to determine explicit conditions characterizing the regions of fast convergence. The difficulty here is two-fold. First, introducing biases requires that one solve a two-dimensional diffusion equation. And second, recall that the coefficients of this partial differential equation are determined pointwise by a double integral. Without velocity biases the outer

integral possesses no analytic description. In the presence of some velocity biases, however, even the inner integral is an elliptic integral.

More Complicated Tasks

More work needs to be done on solving tasks using simple feedback loops. As a first step, one should consider the task of attaining a spherical region in n-dimensional space, with n greater than two. It would be interesting to see whether the degradation of convergence times is simply a function of the increased drift of randomized strategies in higher dimensional spaces, or whether sensing degrades as well. Another direction to explore is the solution of tasks in which line-of-sight distance is not a good progress measure. One question is whether is is possible to use distance to the goal as a progress measure. If the path to the goal bends a lot, it may be impossible to guarantee progress. Finally, a third direction is the exploration of more complicated sensors and sensor models than those assumed in this thesis. Symmetric error balls are not always the best approximation to the error in a sensor. Said differently, using an error ball may be overly conservative.

Diffusion Approximation

More work is required in the modelling of simple feedback loops. Of particular interest is the extent to which diffusion approximations to randomized strategies are possible. An important criterion is the reliability of predictions based on these approximations.

6.3.3 Learning

We showed through the peg-in-hole implementation and its various abstractions that a system can compensate for sensing biases by randomization. The system employed a simple randomized feedback loop. If one permits the system to retain some history then it can actually learn from its observations, and obtain an estimate of the bias. This estimate may then be used to improve performance. A Kalman filter is one approach for retaining history and obtaining an estimate of the bias. However, one can imagine weaker approaches that do not put as much faith in their estimates. A weaker approach might try to follow the philosophy of preparing for worst-case scenarios. This is a philosophy that underlies the guaranteed-planning approaches and that also underlies the decision by which a simple feedback loop makes progress. A possible learning approach might consist of simply recognizing that convergence tends to be fast from certain regions in state space. In other words, no explicit estimate is made of the sensing bias. Rather, it is estimated indirectly, by delineating certain regions that might serve as subgoals, since convergence from them is probably quick. In this case we are really talking about history across multiple iterations of a strategy as opposed to history within a strategy, though both are possible. Much work remains to be done in learning based on randomization.

6.3.4 Solution Sensitivity

Randomization may be thought of as a perturbation in the space of task solutions. By randomizing, a system hopes to find a solution that matches the unknown initial conditions of the world. An interesting inverse problem is to determine the manner in which a task solution must change in order to remain applicable as one perturbs the initial conditions of the system. It seems that there are critical values of uncertain parameters at which task solutions change drastically. Randomization offers a means of retaining an inapplicable solution by perturbing around this solution. However, the perturbations may have to be great. Whether the nature of the perturbation required to solve the task can be inferred from a study of the sensitivity of task solutions to task parameters is an interesting and open question. Answering this question is likely also to further advance the characterization of task solvability and strategy scope.

Bibliography

[AD] **Aldous, D., and Diaconis, P.** 1986. Shuffling Cards and Stopping Times. *American Mathematical Monthly.* **93**:333–348.

[AKLLR] **Aleliunas, R., Karp, R., Lipton, R., Lovász, L., and Rackoff, C.** 1979. Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. *Proceedings of the Twentieth Annual Symposium on the Foundations of Computer Science*, pp. 218–223.

[Arkin] **Arkin, R. C.** 1989. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research.* 8(4): 92–112.

[BL] **Barraquand, J., and Latombe, J.-C.** 1989. Robot Motion Planning: A Distributed Representation Approach. Report No. STAN-CS-89-1257. Stanford University, Department of Computer Science.

[Bell] **Bellman, R.** 1957. *Dynamic Programming.* Princeton: Princeton University Press.

[Bert] **Bertsekas, D. P.** 1987. *Dynamic Programming: Deterministic and Stochastic Models.* Englewood Cliffs, N.J.: Prentice-Hall.

[BRPM] **Boothroyd, G., Redford, A. H., Poli, C., and Murch, L. E.** 1972. Statistical Distributions of Natural Resting Aspects of Parts for Automatic Handling. *Manufacturing Engineering Transactions.* **1**:93–105.

[BHJLM] **Brady, M., Hollerbach, J. M., Johnson, T. L., Lozano-Pérez, T., and Mason, M. T. (eds).** 1982. *Robot Motion: Planning and Control.* Cambridge, Mass.: MIT Press.

[Briggs] **Briggs, A. J.** 1988. An Efficient Algorithm for One-Step Planar Compliant Motion Planning with Uncertainty. Cornell University, Department of Computer Science.

[Brooks82] **Brooks, R. A.** 1982. Symbolic Error Analysis and Robot Planning. *International Journal of Robotics Research.* **1**(4):29–68.

[Brooks83] **Brooks, R. A.** 1983. Solving the Find-Path Problem by Good Representation of Free Space. *IEEE Transactions on Systems, Man, and Cybernetics.* **SMC–13**(3):190–197.

[BLP] **Brooks, R. A., and Lozano-Pérez, T.** 1985. A Subdivision Algorithm in Configuration Space for Findpath with Rotation. *IEEE Transactions on Systems, Man, and Cybernetics.* **SMC–15**(2):224–233.

[Brost85] **Brost, R. C.** 1985. Planning Robot Grasping Motions in the Presence of Uncertainty. CMU-RI-TR-85-12. Carnegie-Mellon University.

[Brost86] **Brost, R. C.** 1986. Automatic Grasp Planning in the Presence of Uncertainty. *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, California, pp. 1575–1581.

[Brown] **Brown, R. G.** 1983. *Introduction to Random Signal Analysis and Kalman Filtering.* New York: John Wiley and Sons.

[Buc] **Buckley, S. J.** 1987. Planning and Teaching Compliant Motion Strategies. AI-TR-936. Ph.D. Thesis. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

[Caine] **Caine, M. E.** Chamferless Assembly of Rectangular Parts in Two and Three Dimensions. S.M. thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, 1985.

[Can88] **Canny, J. F.** 1988. *The Complexity of Robot Motion Planning.* Cambridge, Mass.: MIT Press.

[Can89] **Canny, J. F.** 1989. On Computability of Fine Motion Plans. *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 177–182.

[CD] **Canny, J., and Donald, B.** 1987. Simplified Voronoi Diagrams. *Proceedings ACM Symposium on Computational Geometry*, Waterloo, pp. 153–161.

[CR] **Canny, J. F., and Reif, J. H.** 1987. New Lower-Bound Techniques for Robot Motion Planning Problems. *Proceedings, 28th Symposium on the Foundations of Computer Science.*

[CRRST] **Chandra, A. K., Raghavan, P., Ruzzo, W. L., Smolensky, R., and Tiwari, P.** 1989. The Electrical Resistance of a Graph Captures its Commute And Cover Times. *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pp. 574–586.

[Desai] **Desai, R.S.** 1987. On Fine Motion in Mechanical Assembly in the Presence of Uncertainty. Ph.D. thesis, University of Michigan, Department of Mechanical Engineering.

[Don84] **Donald, B. R.** 1984. Motion Planning with Six Degrees of Freedom. AI-TR-791. S.M. thesis. Cambridge, Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[Don87a] **Donald, B. R.** 1987. A Search Algorithm for Motion Planning with Six Degrees of Freedom. *Artificial Intelligence.* **31**(3):295–353.

[Don87b] **Donald, B. R.** 1987. Error Detection and Recovery for Robot Motion Planning with Uncertainty. AI-TR-982. Ph.D. thesis. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

[Don89] **Donald, B. R.** 1989. *Error Detection and Recovery in Robotics.* Lecture Notes in Computer Science, No. 336. Berlin: Springer-Verlag.

[Drake] **Drake, S. H.** 1977. Using Compliance in Lieu of Sensory Feedback for Automatic Assembly. Sc.D. thesis. Massachusetts Institute of Technology, Department of Mechanical Engineering.

[DL] **Dufay, B., and Latombe, J.** 1984. An Approach to Automatic Robot Programming Based on Inductive Learning. *Robotics Research: The First International Symposium*, eds. M. Brady, and R. Paul. Cambridge, Mass.: MIT Press, pp. 97–115.

[DynYush] **Dynkin, E. B., and Yushkevich, A. A.** 1969. *Markov Processes: Theorems and Problems.* New York: Plenum Press.

[Erd84] **Erdmann, M. A.** 1984. On Motion Planning with Uncertainty. AI-TR-810. S.M. thesis. Cambridge, Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[Erd86] **Erdmann, M. A.** 1986. Using Backprojections for Fine Motion Planning with Uncertainty. *International Journal of Robotics Research.* **5**(1):19–45.

[ELP] **Erdmann, M., and Lozano-Pérez, T.** 1987. On Multiple Moving Objects. *Algorithmica* (Special Issue on Robotics). **2**(4):477–521.

[EM] **Erdmann, M., and Mason, M. T.** 1988. An Exploration of Sensorless Manipulation. *IEEE Journal of Robotics and Automation.* **4**(4):369–379.

[Ernst] **Ernst, H. A.** 1961. MH-1, A Computer-Operated Mechanical Hand. Sc.D. thesis. Massachusetts Institute of Technology, Department of Electrical Engineering.

[FellerI] **Feller, W.** 1968. *An Introduction to Probability Theory and Its Applications. Volume I.* Third edition. New York: John Wiley and Sons.

[FellerII] **Feller, W.** 1971. *An Introduction to Probability Theory and Its Applications. Volume II.* Second edition. New York: John Wiley and Sons.

[FWY] **Fortune, S., Wilfong, G., and Yap, C.** 1986. Coordinated Motion of Two Robot Arms. *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pp. 1216–1223.

[Gill] **Gill, John.** 1977. Computational Complexity of Probabilistic Turing Machines. *Siam Journal of Computing.* **6**(4):675–695.

[GJ] **Göbel, F., and Jagers, A. A.** 1974. Random Walks on Graphs. *Stochastic Processes and their Applications.* **2**:311–336.

[Goldberg] **Goldberg, K.** 1989. Probabilistic Grasping Strategies. Ph.D. proposal, Carnegie Mellon University, School of Computer Science.

[GB] **Grossman, D. D., and M. W. Blasgen.** 1975. Orienting Mechanical Parts by Computer-Controlled Manipulator. *IEEE Transactions on Systems, Man, and Cybernetics.* 561–565.

[HJW] **Hopcroft, J. E., Joseph, D. A., and Whitesides, S. H.** 1982. On the Movement of Robot Arms in 2-Dimensional Bounded Regions. TR 82-486. Cornell University, Department of Computer Science.

[HSS] **Hopcroft, J. E., Schwartz, J. T., and Sharir, M.** 1984 On the Complexity of Motion Planning for Multiple Independent Objects; *PSPACE*-Hardness of the "Warehouseman's Problem." *International Journal of Robotics Research.* **3**(4):76–88.

[HW84] **Hopcroft, J. E., and Wilfong, G. T.** 1984. Reducing Multiple Object Motion Planning to Graph Searching. Technical Report No. 84-616. Ithaca, N.Y.: Cornell University, Computer Science Dept.

[HW86] **Hopcroft, J. E., and Wilfong, G. T.** 1986. Motion of Objects in Contact. *International Journal of Robotics Research.* **4**(4):32–46.

[Inoue] **Inoue, H.** 1974 (Aug.). Force Feedback in Precise Assembly Tasks. A.I. Memo 308. Cambridge, Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory. Reprinted in Winston, P. H., and Brown, R. H., eds. 1979. *Artificial Intelligence: An MIT Perspective.* Cambridge, Mass.: MIT Press.

[KZ] **Kant, K., and Zucker, S. W.** 1986. Toward Efficient Trajectory Planning: The Path-Velocity Decomposition. *International Journal of Robotics Research.* **5**(3):72–89.

[KT1] **Karlin, S., and Taylor, H. M.** 1975. *A First Course in Stochastic Processes.* Second Edition. New York: Academic Press.

[KT2] **Karlin, S., and Taylor, H. M.** 1981. *A Second Course in Stochastic Processes.* New York: Academic Press.

[Khatib] **Khatib, O.** 1986. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *International Journal of Robotics Research.* **5**(1):90–98.

[KCL] **Khatib, O., Craig, J., and Lozano-Pérez, T.** 1989. *The Robotics Review 1*. Cambridge, Mass.: MIT Press.

[KGV] **Kirkpatrick, S., Gelatt, C. D., Jr., and Vecchi, M. P.** 1983. Optimization by Simulated Annealing. *Science.* **220**(4598):671–680.

[Kodit] **Koditschek, D. E.** 1987. Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations. *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, North Carolina, pp. 1–6.

[KR] **Koditschek, D. E, and Rimon, E.** 1988. Robot Navigation Functions on Manifolds with Boundary. Internal Report, Yale University, Department of Electrical Engineering.

[Koutsou] **Koutsou, A.** 1986. Planning Motion in Contact to Achieve Parts Mating. Ph.D. thesis, University of Edinburgh, Department of Computer Science.

[Lat] **Latombe, J.-C.** 1988. Motion Planning with Uncertainty: The Preimage Backchaining Approach. Report No. STAN-CS-88-1196. Stanford University, Department of Computer Science.

[LauTh] **Laugier, C., and Théveneau, P.** 1986. Planning Sensor-Based Motions for Part-Mating Using Geometric Reasoning Techniques. *Proceedings of the European Conference on Artificial Intelligence*, Brighton, U.K.

[Loz76] **Lozano-Pérez, T.** 1976. The Design of a Mechanical Assembly System. AI-TR-397. S.M. thesis. Cambridge, Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory. Reprinted in part in Winston, P. H., and Brown, R. H., eds. 1979. *Artificial Intelligence: An MIT Perspective*. Cambridge, Mass.: MIT Press.

[Loz81] **Lozano-Pérez, T.** 1981. Automatic Planning of Manipulator Transfer Movements. *IEEE Transactions on Systems, Man, and Cybernetics.* **SMC–11**(10):681–698. Reprinted in Brady, M., et al., eds. 1982. *Robot Motion*. Cambridge, Mass.: MIT Press.

[Loz83] **Lozano-Pérez, T.** 1983. Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers.* **C-32**(2):108–120.

[Loz86] **Lozano-Pérez, T.** 1986. A Simple Motion Planning Algorithm for General Robot Manipulators. A.I. Memo 896. Cambridge, Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[LMT] **Lozano-Pérez, T., Mason, M. T., and Taylor, R. H.** 1984. Automatic Synthesis of Fine-Motion Strategies for Robots. *International Journal of Robotics Research.* **3**(1):3–24.

[LPW] **Lozano-Pérez, T., and Wesley, M.** 1979. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *Communications of the ACM.* **22**(10):560–570.

[MW] **Mani, M., and W. Wilson.** 1985. A Programmable Orienting System for Flat Parts. *Proc., NAMRI XIII.*

[Mas81] **Mason, M. T.** 1981. Compliance and Force Control for Computer Controlled Manipulators. *IEEE Transactions on Systems, Man, and Cybernetics.* **SMC-11**(6):418–432. Reprinted in Brady, M., et al., eds. 1982. *Robot Motion.* Cambridge, Mass.: MIT Press.

[Mas82a] **Mason, M. T.** 1982. Manipulator Grasping and Pushing Operations. AI-TR-690. Ph.D. thesis. Cambridge, Mass.: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[Mas82b] **Mason, M. T.** 1982. Compliant Motion. *Robot Motion,* eds. M. Brady, et al. Cambridge, Mass.: MIT Press, pp. 305–322.

[Mas84] **Mason, M. T.** 1984. Automatic Planning of Fine-Motions: Correctness and Completeness. *Proceedings of the 1984 IEEE Int. Conf. on Robotics and Automation*, pp. 492–503.

[Mas85] **Mason, M. T.** 1985. The Mechanics of Manipulation. *Proceedings of the 1985 IEEE Int. Conf. on Robotics and Automation*, pp. 544–548.

[Mas86] **Mason, M. T.** 1986. Mechanics and Planning of Manipulator Pushing Operations. *International Journal of Robotics Research.* **5**(3):53–71.

[Montroll] **Montroll, E. W.**, 1964. Random Walks on Lattices. *Proceedings of Symposia in Applied Mathematics,* pp. 193–220.

[Nat86] **Natarajan, B. K.** 1986. An Algorithmic Approach to the Automated Design of Parts Orienters. *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pp. 132–142.

[Nat88] **Natarajan, B. K.** 1988. The Complexity of Fine Motion Planning. *International Journal of Robotics Research.* **7**(2):36–42.

[NWD] **Nevins, J., Whitney, D., Drake, S., Killoran, D., Lynch, M., Seltzer, D., Simunovic, S., Spencer, R. M., Watson, P., and Woodin, A.** 1975. Exploratory Research in Industrial Modular Assembly. Report R-921. C.S. Draper Laboratory, Cambridge, Mass.

[ODSY] **Ó'Dúnlaing, C., Sharir, M., and Yap, C.** 1982. Retraction: A New Approach to Motion-Planning. NYU-Courant Institute, Robotics Lab Technical Report.

[ODY] **Ó'Dúnlaing, C., and Yap, C.** 1985. A Retraction Method for Planning the Motion of a Disc. *J. Algorithms*, **6**:104–111.

[OHR] **Ohwovoriole, M. S., Hill, J. W., and Roth, B.** 1980. On the Theory of Single and Multiple Insertions in Industrial Assemblies. *Proc. 10th International Symposium on Industrial Robots.* Bedford, U.K.: IFS Publications, pp. 545–558.

[OR] **Ohwovoriole, M. S., and Roth, B.** 1981. A Theory of Parts Mating for Assembly Automation. *Proceedings of the Robot and Man Symposium 81*, Warsaw, Poland.

[Pap] **Papadimitriou, C. H.** 1985. Games against Nature. *Journal of Computer and System Sciences.* **31**:288-301.

[PT] **Papadimitriou, C. H., and Tsitsiklis, J. N.** 1987. The Complexity of Markov Decision Processes. *Mathematics of Operations Research.* **12**(3):441–450.

[PS] **Paul, R. P., and Shimano, B.** 1976. Compliance and Control. *Proceedings of the 1976 Joint Automatic Control Conference*, pp. 694-699. Reprinted in Brady, M., et al., eds. 1982. *Robot Motion.* Cambridge, Mass.: MIT Press.

[Pesh] **Peshkin, M. A.** 1986. Planning Robotic Manipulation Strategies for Sliding Objects. Ph.D. Thesis. Carnegie-Mellon University, Physics Department.

[RC] **Raibert, M. H., and Craig, J. J.** 1981 (June). Hybrid Position/Force Control of Manipulators. *Journal of Dynamic Systems, Measurement, and Control.* **102**:126–133. Reprinted in Brady, M., et al., eds. 1982. *Robot Motion.* Cambridge, Mass.: MIT Press.

[Reif] **Reif, J. H.** 1979. The Complexity of the Mover's Problem and Generalizations. *Proceedings, 20th Symposium on the Foundations of Computer Science.*

[RS] **Reif, J., and Sharir, M.** 1985 (Portland, Oregon). Motion Planning in the Presence of Moving Obstacles. *Proceedings, 26th IEEE Symposium on the Foundations of Computer Science*, pp. 144–154.

[Ross] **Ross, S. M.** 1983. *Stochastic Processes.* New York: John Wiley and Sons.

[Sal] **Salisbury, J. K.** 1980. Active Stiffness Control of a Manipulator in Cartesian Coordinates. Paper delivered at the *19th IEEE Conference on Decision and Control*, Albuquerque, New Mexico.

[SHS] **Schwartz, J., Hopcroft, J., and Sharir, M.** 1987. *Planning, Geometry, and Complexity of Robot Motion.* New Jersey: Ablex Publishing.

[ScShII] **Schwartz, J. T., and Sharir, M.** 1983. On the Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Advances in Applied Mathematics.* **4**:298–351.

[ScShIII] **Schwartz, J. T., and Sharir, M.** 1983. On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Amidst Polygonal Barriers. *International Journal of Robotics Research.* **2**(3):46–75.

[SY] **Schwartz, J., and Yap, C.** 1986. *Advances in Robotics.* Hillside, New Jersey: Lawrence Erlbaum Associates.

[Sim75] **Simunovic, S. N.** 1975 (Sept. 22–24, Chicago, Illinois). Force Information in Assembly Processes. *Proceedings 5th International Symposium on Industrial Robots.* Bedford, U.K.: IFS Publications, pp. 415–431.

[Sim79] **Simunovic, S. N.** 1979. An Information Approach to Parts Mating. Sc.D. thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering.

[SJ] **Sinclair, A., and Jerrum, M.** 1987. Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains. Report CSR-241-87, University of Edinburgh, Department of Computer Science.

[SpY] **Spirakis, P., and Yap, C.** 1984. Strong NP-hardness of Moving Many Discs. *Information Processing Letters.* **19**:55–59.

[Stengel] **Stengel, R. F.** 1986. *Stochastic Optimal Control.* New York: John Wiley and Sons.

[Sturges] **Sturges, R. H., Jr.** 1988. A Three-Dimensional Assembly Task Quantification with Application to Machine Dexterity. *International Journal of Robotics Research.* **7**(4):34–78.

[Tay] **Taylor, R. H.** 1976. A Synthesis of Manipulator Control Programs from Task-Level Specifications. AIM–282. Ph.D. thesis. Stanford, Calif.: Stanford University, Artificial Intelligence Laboratory.

[TMG] **Taylor, R. H., Mason, M. T., and Goldberg, K. Y.** 1987. Sensor-Based Manipulation Planning as a Game with Nature. *Proceedings, Fourth International Symposium of Robotics Research.*

[Turk] **Turk, M. A.** 1985 (Sept., Cambridge, Ma.). A Fine-Motion Planning Algorithm. *SPIE Intelligent Robots and Computer Vision*, pp. 113–120.

[Udupa] **Udupa, S. M.** 1977. Collision Detection and Avoidance in Computer Controlled Manipulators. Ph.D. thesis. California Institute of Technology, Department of Electrical Engineering.

[Valade] **Valade, J.** 1984. Automatic Generation of Trajectories for Assembly Tasks. *Proceedings of the Sixth European Conference on Artificial Intelligence*, Pisa, Italy.

[Wang] **Wang, Yu.** 1989. Dynamic Analysis and Simulation of Mechanical Systems with Intermittent Constraints. Ph.D. thesis, Carnegie Mellon University, Department of Mechanical Engineering.

[WM] **Wang, Yu, and Mason, M.** 1987. Modeling Impact Dynamics for Robotic Operations. *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, pp. 678–685.

[Whit77] **Whitney, D. E.** 1977. Force Feedback Control of Manipulator Fine Motions. *Journal of Dynamic Systems, Measurement, and Control.* **98**:91–97.

[Whit82] **Whitney, D. E.** 1982. Quasi-Static Assembly of Compliantly Supported Rigid Parts. *Journal of Dynamic Systems, Measurement, and Control.* 104:65–77. Reprinted in Brady, M., et al., eds. 1982. *Robot Motion.* Cambridge, Mass.: MIT Press.

[WG] **Will, P. M., and Grossman, D. D.** 1975. An Experimental System for Computer Controlled Mechanical Assembly. *IEEE Transactions on Computers.* **C-24**(9):879–888.

[Yap84] **Yap, C. K.** 1984. Coordinating the Motion of Several Discs. Technical Report No. 105. New York University Computer Science Department, Courant Institute of Mathematical Sciences.

[Yap86] **Yap, Chee K.** 1986. Algorithmic Motion Planning. In *Advances in Robotics*, ed. J. Schwartz and C. Yap. Hillside, New Jersey: Lawrence Erlbaum Associates.

[Z] **Zuckerman, D.** 1989. Covering Times of Random Walks on Bounded Degree Trees and Other Graphs. Internal Report, University of California, Berkeley, Computer Science Division.